A level Computer Science Study Pack:1

Paper 1 Exam preparations

Marega T
MBIZO HIGH SCHOOL 2018

PAPER 1

TOPIC	**WEIGHTING (30%)**	**Number of Qxns**	**Marks**
Data Representation	15	1 to 2	15
Networking	20	2	20
Computer Architecture	20	2	20
Security and Ethics	10	1	20
Algorithm Design and Data Structures	15	1 to 2	20
Databases	10	1	10
Enterprising	10	1	10
TOTAL	**100**	**10 to 12**	**100**

SKILL	**Paper 1(%)**	
Knowledge and Understanding	30	(3 to 4 qxns)
Problem Solving	50	(5 to 6 qxns)
Practical Skills	20	(2 qxns)
TOTAL	**100**	**10 to 12 qxns**

Paper 1: Theory (100 Marks)

The paper consists of 10 to 12 compulsory questions.

1 **Define** is intended literally, only a formal statement or equivalent paraphrases being required.

2 **State** implies a concise answer with little or no supporting argument e.g. numerical answer that can readily be obtained by inspection

3 **List** requires a number of points generally each of one word with no elaboration, where a number of points is specified this should not be exceeded.

4 **Explain** may imply reasoning or some reference to theory depending on the context

5 **Describe** requires the candidate to state in words (using diagrams where appropriate) the main points of the concept

6 **Outline** implies brevity that is restricting the answer to given essentials

7 **Predict/deduce** the candidate is expected to produce the expected answer by making a logical connection between other pieces of information

8 **Suggest** it is used in two main contexts that is either to imply that there is no unique answer or to imply that learners are expected to apply their general knowledge

9 **Find** is a general term that may variously be interpreted as *calculate, measure, determine* etc

10 **Determine** often implies that the quantity concerned cannot be measured directly but is obtained by calculation

Data Representation

1. convert number bases
2. multiply and divide binary numbers
3. normalize floating point binary numbers
4. represent data and character sets
5. distinguish arithmetic errors
6. interpret arithmetic errors

Networking

1. compare OSI and TCP/IP models
2. explain the format of an IP address
3. distinguish between public and private IP addresses
4. explain the role of DNS
5. describe Routing Information Protocol(RIP) and Open Shortest Path First (OSPF) routing protocols
6. differentiate cloud service models
7. describe cloud types

Computer Architecture

1. explain the principle of operation of passive and active electronic components
2. describe the Von Neumann and Harvard Architecture
3. explain the use of buses
4. demonstrate the use of logic gates
5. describe the functions of processor components
6. identify factors affecting processor speeds
7. explain the importance of pipelining
8. describe the fetch-decode execute cycle
9. identify types of interrupts
10. justify why computers use interrupts
11. explain addressing modes
12. construct an architectural design for a given scenario

Security and Ethics

1. explain the difference between data privacy and integrity
2. analyse common threats and vulnerabilities on computer systems
3. identify sources of vulnerability
4. describe how data is kept safe during storage and transmission
5. evaluate tools used to eliminate vulnerabilities
6. identify relevant ICT legislative and regulatory frameworks
7. identify risks to Computer Systems
8. explore techniques and practices of risk management
9. outline the importance of securing data at off-site locations (cloud computing)
10. identify code of ethics and professional practices in the Computing field

11. discuss security policies, laws and computer crime
12. identify environmental impact from computer use and disposal
13. analyse the ethical impact of existing and emerging technologies
14. explain the attributes of business ethics

Algorithm Design and Data Structures

1. analyse algorithms for a given situation
2. design algorithms for a given situation
3. demonstrate familiarity with standard algorithms
4. distinguish between dynamic and static data structures
5. perform operations on binary trees and arrays
6. outline primitive data types
7. use example to demonstrate recursion

Databases

1. describe the file based approach database systems
2. outline features of a DBMS
3. describe features of relational database which address limitations of a file based approach
4. write SQL commands
5. develop interfaces and queries using DBMS tools
6. access data in a database through a high level language
7. design database applications using Entity relationship diagrams (ERDs)
8. convert ERDs to relational databases schema in standard normal form
9. normalise database tables up to second normal form

Enterprising

1. identify areas where computer science is applied
2. evaluate the importance of e- Business
3. explain the elements of marketing
4. design an ICT related business proposal
5. describe the role of Intellectual Property Rights
6. analyse global trends in the field of computing

Data Representation

1. Convert Number Bases

(a) Conversion from decimal (denary) to Binary

Divide the denary number by 2, listing the remainders until the answer is 0 remainder 1. Take the remainders only from the last one until the first.

For example:

2	20	
2	10	r 0
2	5	r 0
2	2	r 1
2	1	r 0
	0	r 1

The answer is therefore 10100_2

Or 00010100 in 8 bits

Or. Use the binary powers to represent to determine the representation e.g 20 in binary

Power	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Equivalent to:	128	64	32	16	8	4	2	1
Binary Digits	0	0	0	1	0	1	0	0

This means $20 = 0x2^7 + 0x2^6 + 0x2^5 + 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 0x2^0$

$= 16 + 4$

(b) Binary to decimal conversion

Raise each bit to its binary power equivalent, from right going to the left, starting at 20

Power	2^4	2^3	2^2	2^1	2^0
Equivalent to:	16	8	4	2	1
Binary Digits	1	0	1	0	0

Add all the equivalent to values in the table, whose binary digit correspond to 1 and add them.

The result is the denary equivalent.

That is 16 + 4 = 20. This is a short form of (16 x 1) + (4 x1) = 20

(c) Decimal to octal

- Octal number contains only digits from 0 to 7.
- As on binary, take the number, divide it by 8 and take the remainders only, e.g. 78 to octal will be expressed as:

8	78	
8	9	r 6
8	1	r 1
	0	r 1

$= 116_8$

Or. Use the binary grid to represent to determine the representation, convert the decimal number to *binary*, group the bits into 3s starting from your right, then convert the grouped bits to denary

Power	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary Digits	0	1	0	0	1	1	1	0

2^2	2^1	2^0	2^2	2^1	2^0	2^2	2^1	2^0
0	0	1	0	0	1	1	1	0
	1			1			6	

$= 116$

maregat@mbizohighschool.ac.zw

(d) Octal to decimal

Power	8^2	8^1	8^0
Equivalent to:	64	8	1
Octal Digits	1	1	6

= (64 x 1) + (8 x 1) + (1 x 6) = **78**

Or. Use the binary powers to represent to determine the representation, convert each octal digit to *binary*, group the bits, then convert the grouped bits to denary

Octal	1			1			6		
Binary	0	0	1	0	0	1	1	1	0
Powers	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Denary	$1X2^6 + 1X2^3 + 1X2^2 + 1X2^1 = 64+8+4+2$ =**78**								

(e) Decimal to hexadecimal

- A hexadecimal number contains numbers from 0 to 15 with numbers 10 to 15 being represented by uppercase alphabetic characters from A to F respectively.

Decimal Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal Equivalent	0	1	2	3	4	5	6	7	8	9	**A**	**B**	**C**	**D**	**E**	**F**

As on binary, take the number, divide it by 16 and take the remainders only, e.g. 209 to hexadecimal will be expressed as:

16	209
	13 r 1
	0 r 13

This gives us (13)1, but 13 is represented as D in our table above. Thus the answer will be **$D1_{16}$**

Or. Use the binary powers to represent to determine the representation, convert the decimal number to *binary*, group the bits into 4s starting from your right, then convert the grouped bits to denary

Power	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary	1	1	0	1	0	0	0	1

2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
1	1	0	1	0	0	0	1
			13				1

= D1

(f) Hexadecimal to decimal

Power	16^1	16^0
Equivalent to:	16	1
Octal Digits	13 (D)	1

= (16 x 13) + (1 x 1) = **209**

Or. Use the binary powers to represent to determine the representation, convert each hexadecimal digit to *binary*, group the bits, and then convert the grouped bits to denary

hexadecimal	13(D)				1			
Binary	1	1	0	1	0	0	0	1
Powers	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Denary	$1X2^7 + 1X2^6 + 1X2^4 + 1X2^0 = 128+64+16+1$ =**209**							

maregat@mbizohighschool.ac.zw

2. Multiply And Divide Binary Numbers

Binary multiplication

- is actually much simpler to calculate than decimal multiplication because in binary multiplication, we only need to remember the following,

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

***Note** that since binary operates in base 2, the multiplication rules we need to remember are those that involve 0 and 1 only. As an example of binary multiplication we have 101 times 11,

101
x11

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.

101
x11
101
1010 <-- the 0 here is the placeholder

The next step, as with decimal multiplication, is to add. The results from our previous step indicates that we must add 101 and 1010, the sum of which is 1111.

101
x11
101
1010
1111

Binary Division

- not as difficult as it may seem, make use of the long division method

Let's take for example the division of 11 into 1011

```
      11   R=10
11 | 1011
     -11
      101
      -11
       10 <-- remainder, R
```

To check our answer, we first multiply our divisor 11 by our quotient 11. Then we add its' product to the remainder 10, and compare it to our dividend of 1011.

Binary Addition

Addition without a Carry

Binary Addition	Decimal Addition
1 0 + 0 1 Result 1 1	2 + 1 3
$11_2 = 3_{10}$	

Addition with Carry

Binary Addition	Decimal Addition
1 1 ← Carry 0 1 + 1 1 Result 1 0 0	1 + 3 4
$100_2 = 4_{10}$	

Binary Addition	Decimal Addition
1 1 1 1 1 ← Carry 1 0 1 1 1 + 1 1 1 0 0 1 1 1 1 0 1 1 0	2 3 + 2 8 3 5 4
$110110_2 = 54_{10}$	

Binary subtraction

Subtraction of binary numbers follow that you add the negative number to a positive number ie. 7-6 is the same as 7 + (-6).

⇨ 000111
\- 000110

⇨ 000111 (7)
\+ 111010 (-6)
1 000001
overflow

Find the complement/ the negative of the number being subtracted
Convert 0s to 1s and add 1
000110 => 111001
\+ 1
111010 (**-6**)

Signed and Unsigned Numbers

A binary number may be positive("+") or negative("-").However, binary numbers use 0 (for positive) and 1 (for negative) in the computer. An *signed binary number* consists of two parts: sign bit and magnitude. The left most bit (Most Significant Bit (MSB)) is the *sign bit*.

Sign and Magnitude representation

01100011 is a positive number since its sign bit is 0
11001011 is a negative number since its sign bit is 1.
An 8-bit *signed number* can represent data in the range -128 to +127 (-2^7 to $+2^7-1$).

The magnitude of a number is its natural value, regardless of the sign. Thus the magnitude of -25 and +25 is 25 (not considering the sign in this situation). In binary form, 25 = **11001**. Thus using Sign and Magnitude representation:

+25 = **0** 11001
-25 = 1 11001

What only differ is the **sign bit**, not the **magnitude**.

Representation of Negative Numbers

Negative numbers are mostly represented using the complement of number. Complement of numbers can be in 1's complement or 2's complement. The complement of a number behaves like the negative of the original number.

1's Complement One's complement of a binary number is obtained by simply converting 1s to 0s and 0s to 1s. For example, given the following four-bit binary number 1010_2, its 1's complement becomes 0101_2. The alternating of bits only applies to negative numbers, positive numbers do not change. For example

+6 = 00000110_2
-6 = 11111001_2

-6 is the complement (negative) of +6. Just convert 1s to 0s and 0s to 1s and thus -6 in 1's complement.

Given the 1's complement we can find the magnitude of the number by taking it's 1's complement. The range of numbers that can be represented in 1's complement is found by the formula: $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

If the binary number has 8-bits (n=8). Thus the range of numbers will be from (-127) 10000000_2 to 01111111_2 (127)
Therefore the largest number that can be represented in 8-bit 1's complement is = 127. The smallest is -127.
However 1's complement has a problem that it has two different representations (values) for zero, which are 00000000_2 and 11111111_2 both represent zero.

When **adding** binary numbers using 1's complement, the carry bit is added back to the sum in the rightmost position. There is **no overflow** as long as the magnitude of the result is not greater than 2^{n1}- 1 . **We do not throw away the carry bit.**

2'S COMPLEMENT

Two's complement of number is obtained by:

a) Positive numbers remain the same
b) Negative numbers:
   - Change the number to its 1's complement.
   - Add 1 to the result and the number will be in 2's complement.

OR

   - Rewrite the bits starting from the right hand side, all 0s take as they are at their respective position and the first 1 value encountered. The rest alternate a 1 to 0 and a 0 to a 1 and your number will be in 2's complement.

For example, in 2's complement,
+6 = 00000110_2
-6 = 11111010_2

We can also find the magnitude the 2's complement number. The largest number that can be represented in 8-bit 2s complement is 01111111_2 = 127. The smallest is 10000000_2 = -128.

Converting From Binary Two's Complement To Denary

	1 1 1 1 1 0 1 1
Step-1	0 0 0 0 0 1 0 0

Complement the number.

Step-2	- 0 0 0 0 0 1 0 1

Add one add prefix a minus sign.

Step-3	- 5

Convert binary to decimal.

When the addition of two values results in a carry, the carry bit is ignored and is thrown away. There is no **overflow** as long as the magnitude is not greater than 2^{n-1}-1 nor less than $-(2^{n-1})$.
The two's-complement system has the advantage that the fundamental arithmetic operations of addition, subtraction, and multiplication are identical to those for unsigned binary numbers. This property makes the system both simpler to implement and capable of easily handling higher precision arithmetic. Also, zero has only a single representation, other than in ones'-complement where it has two values.

3. Normalize Floating Point Binary Numbers

Fixed Point integer representation

The decimal point in an integer is implied and does not change its position. There is no memory space for the decimal point. However computers represent a finite number of digits. This limitation allows us to evaluate the maximum and minimum possible numbers that can be represented. These include:

- **Maximum Positive Number:-**

  Positive numbers start with a 0, thus 01111111 = +127

- **Minimum Positive Number:-**

  This evaluates to 00000001 = +1

- **Smallest Magnitude Negative Number:-**

  Negative numbers start with a 1. This gives us 11111111 = -1

- **Largest Magnitude Negative Number:-**

  This is 10000000 = -128

The overall range of numbers here is -2^{n-1} to $(2^{n-1}-1)$

Positive Fixed Point fractional Binary Numbers

- **Maximum Positive Number:-**

  This becomes 0.1111111 = $1-{}^{1}/_{128}$ = 0.9921875

- **Minimum Positive Number:-**

  This evaluates to 0.0000001 = $1/2^7$=0.0078125

- **Smallest Magnitude Negative Number:-**

  Negative numbers start with a 1. This gives us 1.1111111 = $-1/2^7$=0.0078125

- **Largest Magnitude Negative Number:-**

  This is 1.0000000 = -1

The decimal point is fixed at one position and therefore does not move. In binary we can have functional column headings.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}
128	64	32	16	8	4	2	1		0.5	0.25
1	0	1	0	0	1	0	0	.	1	1

=164.75

Binary Fraction	Fraction	Decimal
0.1	1/2	0.5
0.01	1/4	0.25
0.001	1/8	0.125
0.0001	1/16	0.0625

Advantages and Disadvantages Of The Fixed Point Binary System

- Preserves accuracy as required and is Easy to convert.
- However it has a limited degree of accuracy.

FLOATING POINT ARITHMETIC

A fixed point notation (e.g. two's complement) allows a range of positive and negative integers centred around 0 to be represented

- By assuming a fixed binary or radix point, this format would allow numbers with a fractional component to be represented
- But – this approach has limitations – **very large numbers or very small fractions cannot be represented**
- Floating point representation allows us to represent very large numbers and very small fractions

The *floating point number representation* uses two registers. The first register stores the number without the binary point. The second register stores a number that indicates the position of the binary point in the first register.

In decimal notation the number 23.456 can be written as 0.23456×10^2. This means that we need only store, in decimal notation, the numbers 0.23456 and 2. The number 0.23456 is called the *mantissa* and the number 2 is called the *exponent*. This is what happens in binary. Similarly, in decimal, 0.0000246 can be written 0.246×10^{-4}. Now the mantissa is 0.246 and the exponent is –4.

The floating point representation of a number has two parts: **mantissa** and **exponent**. The mantissa is a signed **fixed point number**. The exponent shows the **position of the binary point** in the mantissa. For example, the binary number +11001.11 with an 8-bit mantissa and 6-bit exponent is represented as follows -

• Mantissa is 01100111. The left most 0 indicates that the number is positive.

• Exponent is 000101. This is the binary equivalent of decimal number +5.

• The floating point number is Mantissa x 2^{exponent} , i.e. + (.1100111) x 2^{+5}.

Example: consider the binary number 10111. This could be represented by 0.10111×2^5 or 0.10111×2^{101}. Here 0.10111 is the mantissa and 101 is the exponent.

Thus, in binary, 0.00010101 can be written as 0.10101×2^{-11} and 0.10101 is the mantissa and –11 is the exponent.

It is now clear that we need to be able to store two numbers, the mantissa and the exponent. This form of representation is called *floating point form*. Numbers that involve a fractional part, like 2.467_{10} and 101.0101_2 are called *real numbers*.

Give the denary number which would have 01000000 00000000 as its binary, floating point representation in this computer
The answer is 0.5 or ½ because it will be 0.1 x 2^0
It is not possible to represent zero as a normalised floating point number because a normalised value must have the first two bits of the mantissa different. Therefore one must be a 1- which must represent either -1 or + ½, but not zero.

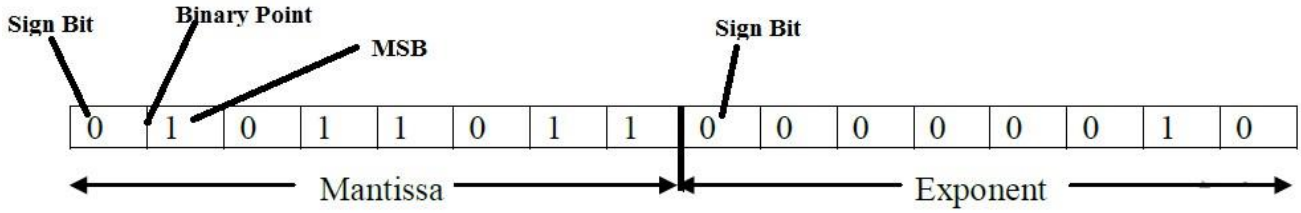
Weaknesses of Floating Point representation

- One number can have different representations, for example, 2 (010.0000000) can be represented as
  0.100000000 x 2^2
  0.010000000 x 2^3
  0.001000000 x 2^4
- Causes errors as some less significant bits are lost, causing some unnecessary errors

Normalisation

This is done to simplify operations and expressing numbers in standard form.

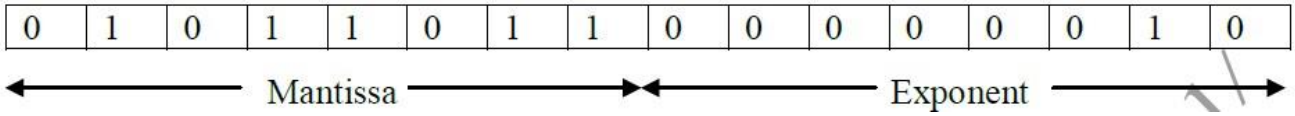
Normalisation Principles:
Let us look at the following diagram



- A number is expressed in two main parts, which are:
  - The **mantissa (or fractional)** part: this is always a fraction. The binary point is always between the Sign bit and the MSB.
  - The **Exponent (or characteristic):** it is always an integer (whole number) and can be positive of negative depending on its sign bit. The left most bit of the exponent is a sign bit.
- The first two digits of the mantissa **must be different** in a normalised number. Thus:
  - If the mantissa is Positive, the Sign Bit is always 0 and the MSB is always 1.
  - If the mantissa is negative, the Sign Bit is always 1 and the MSB is always 0.
- For a negative number, there must be **NO** leading 1s to the left of the MSB, excluding the sign bit.

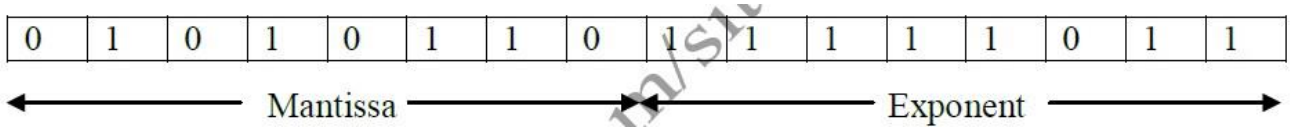
- For a positive number, there must be **NO** leading 0s to the left of the MSB, excluding the sign bit.

- With positive numbers, the binary point in the mantissa was always placed immediately before the first non-zero digit because it allows us to use the maximum number of digits. Suppose we use 8 bits to hold the mantissa and 8 bits to hold the exponent. The binary number
10.11011 becomes 0.1011011×2^{10} and can be held as



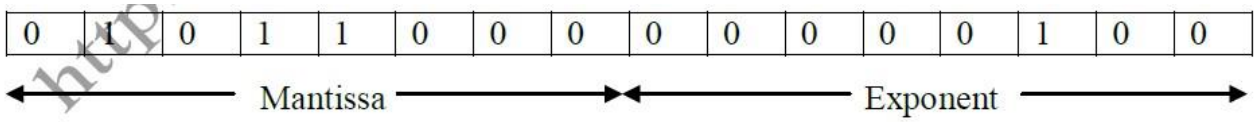
The first digit of the mantissa is zero and the second is one. The mantissa is said to be normalised if the first two digits are different. Thus, for a positive number, the first digit is always zero and the second is always one. The exponent is always an integer and is held in two's complement form.

Now consider the binary number 0.00000101011 which is 0.101011×2^{-101}. Thus the mantissa is 0.101011 and the exponent is –101. Again, using 8 bits for the mantissa and 8 bits for the exponent, we have

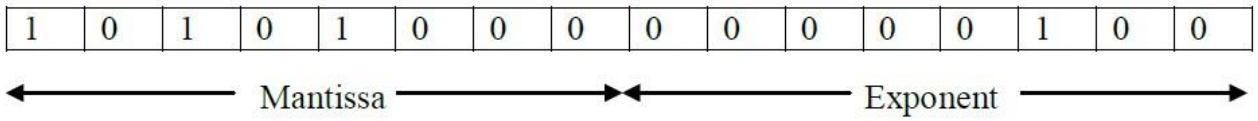


Because the two's complement of –101, using 8 bits, is 11111011.
The reason for normalising the mantissa is in order to hold numbers to as high a degree of accuracy as possible.

Care needs to be taken when normalising negative numbers. The easiest way to normalise negative numbers is to first normalise the positive version of the number. Consider the binary number –1011. The positive version is $1011 = 0.1011 \times 2^{100}$ and can be represented by



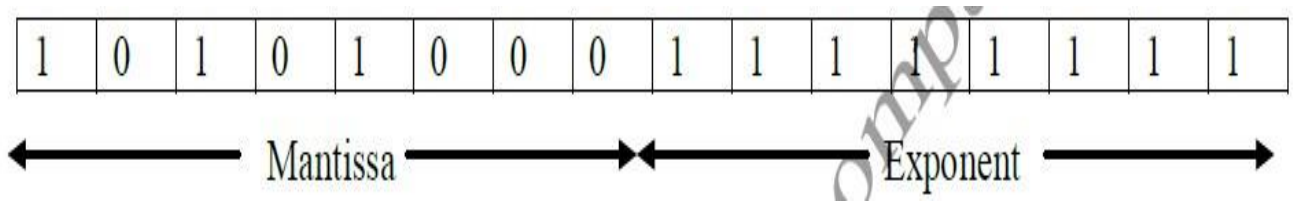
Now find the two's complement of the mantissa and the result is



Notice that the first two digits are different.

As another example, change the decimal fraction –11/32 into a normalised floating point binary number.
Leave the negative sign first and solve it for just $^{11}/_{32}$, $^{11}/_{32} = {}^{1}/_{4} + {}^{1}/_{16} + {}^{1}/_{32} = 0.01 + 0.0001 + 0.00001 = 0.01011$; converted to binary equivalent.

Now we have 8 bits mantissa and 8 bits exponent, 00101100 00000000,
It is not normalized so normalize it by removing a 0 from location worth $^1/_2$ in mantissa and subtracting that 1 location from exponent 0 reveals -1. That is, 01011000 11111111; this is Floating
Point equivalent of $^{11}/_{32}$.
For $^{-11}/_{32}$ keep the exponent same and in mantissa start from right side. Keep all digits same until first 1 and toggle rest.



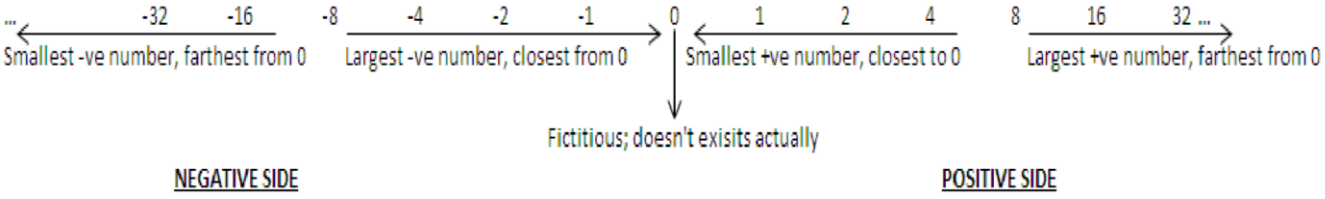
Benefits of Normalisation

- Ensures that a single representation of a number is maintained (standardisation).
- Ensures maximum possible accuracy with a given number of bit is maintained.
- Can be used to detect error conditions such as underflow and overflow
- Tires to maximise the range of numbers that can be represented in a fixed point representation (Range and accuracy is limited in fixed point representation)

Range and Precision/ Accuracy and Range

- **The size of the exponent determines the range of numbers that can be represented**
  - The range of numbers is expanded by increasing the number of bits that are used to represent the exponent. This will however decrease precision.
  - Reducing the number of bits in the exponent will reduce the range because power of two which the mantissa is multiplying by is decreased.
  - At the same time decreasing the exponent's bits will increase accuracy because more digits are represented after the binary point.

- **The size of the significant determines the precision of the numbers that can be represented**
  - Precision can be increased by increasing the number of bits that are used to represent the significant
  - This will decrease the range.
- **The only way to increase both range and precision is to use more bits**
  - That is, the use of single-precision numbers, double-precision numbers, etc



If we use more bits for the mantissa we will have to use fewer bits for the exponent. Let us start off by using 8 bits for the mantissa and 8 bits for the exponent for explanations below:

- The **largest positive value** we can have for the **mantissa** is 0.1111111 and
- The **largest positive number** we can have for the **exponent** is 01111111.
- This means that we have $0.1111111 \times 2^{1111111} = 0.1111111 \times 2^{127}$.
- This means that the largest positive number is almost 1×2^{127}.

Also:

- The **smallest positive mantissa** is 0.1000000 and
- the smallest exponent is 10000000.
- This represents $0.1000000 \times 2^{10000000} = 0.1000000 \times 2^{-128}$ which is very close to zero; in fact it is 2^{-129}.

On the other hand:

- The largest negative number (i.e. the negative number closest to zero) is $1.0111111 \times 2^{10000000} = -0.1000001 \times 2^{-128}$
- We cannot use 1.1111111 for the mantissa because it is not normalised. The first two digits must be different.

Furthermore:

- The smallest negative number (i.e. the negative number furthest from zero) is $1.0000000 \times 2^{01111111} = -1.0000000 \times 2127 = -2^{127}$.

Zero cannot be represented in normalised form. This is because 0.0000000 is not normalised because the first two digits are the same. *A normalised value must have the first two bits of the mantissa different.* Therefore one of them must be a 1 which must represent either -1 or + ½, but not zero. Usually, the computer uses the smallest positive number to represent zero.

Also, size of number mean the furthest to the left on a number line so -1 is a bigger number than -2. Whereas, if we talk about largest magnitude negative number then the -2 is greater magnitude than -1 because the integer value is greater (not considering the sign).

For a positive number n,

$2^{-129} \leq n < 2^{127}$

For a negative number n

$-2^{127} \leq n < -2^{-129}$

4. Represent Data And Character Sets

Character Set - *All the characters that a system can recognise, which often equates to characters on the keyboard*.

Each character is represented using a unique set of bits which are equivalent to 1 or 2 bytes. **Character set of a computer is represented as** binary codes, ASCII, UNICODE and EBCDic using 8 bits.

ASCII - American Standard Code for Information Interchange

It is a set of codes that a computer understands and is represented in a single byte (8 bits) per character.

ASCII codes are of two types –ASCII-7 and ASCII-8.

- ***ASCII-7*** is a 7-bit standard ASCII code. In ASCII-7, the first 3 bits are the zone bits and the next 4 bits are for the digits. ASCII-7 allows 27 = 128 combinations. 128 unique symbols are represented using ASCII-7. ASCII-7 has been modified by IBM to ASCII-8.
- ***ASCII-8*** is an extended version of ASCII-7. ASCII-8 is an 8-bit code having 4 bits for zone and 4 bits for the digit. ASCII-8 allows 28 = 256 combinations. ASCII-8 represents 256 unique symbols.

ASCII is used widely to represent data in computers.

- The ASCII-8 code represents 256 symbols.
  - Codes 0 to 31 represent control characters (non-printable), because they are used for actions like, Carriage return (CR), Bell (BEL) etc.
  - Codes 48 to 57 stand for numeric 0-9.
  - Codes 65 to 90 stand for uppercase letters A-Z.
  - Codes 97 to 122 stand for lowercase letters a-z.
  - Codes 128-255 are the extended ASCII codes.

In the ASCII character set, each binary value between 0 and 127 is given a specific character. Most computers extend the ASCII character set to use the full range of 256 characters available in a byte. The upper 128 characters handle special things like accented characters from common foreign languages.

Binary System

Data is represented in 0s and 1s, thus in base 2. It is obtained by dividing the denary number by 2, taking the remainders only. The number of bits in the answer does not matter unless specified.

BCD

Each decimal digit is represented by its own 4-bit binary code as follows:

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

The number 3765 is thus coded as $0011\ \ 0111\ \ 0110\ \ 0101_2$

BCD is used to represent some numbers that are **not proper numbers** (numbers that don't behave like numbers). A barcode looks like a number, but if the barcodes are added together the result is not a barcode for any product. The arithmetic does not give a sensible answer. Values which look like numbers but do not behave like numbers are often stored in binary coded decimal (BCD). Each digit is simply changed into a four bit binary number which are then placed one after another in order.

- This has the advantage that it is easy to convert a number from BCD to decimal form and vice versa.
- There is no rounding off of numbers when computing fractional numbers, thus no errors due to rounding off.
- Used in businesses where significant digit needs to be retained.

However:

- As compared to pure binary, more bits are needed to store a number, thus more memory is needed
- Calculations with such numbers are more complex than in pure binary numbers

EBCDIC

The Extended Binary Coded Decimal Interchange Code (EBCDIC) uses 8 bits (4 bits for zone, 4 bits for digit) to represent a symbol in the data.

- EBCDIC allows 28 = 256 combinations of bits.
- 256 unique symbols are represented using EBCDIC code. It represents decimal numbers (0 - 9), lower case letters (a-z), uppercase letters (A-Z), Special characters, and Control characters (printable and non-printable e.g. for cursor movement, printer vertical spacing etc.).
- EBCDIC codes are used, mainly, in the mainframe computers.

UNICODE

Unicode is a universal character encoding standard for the representation of text which includes letters, numbers and symbols in multi-lingual environments. This is an international 16-bit data coding method which represents 65536 different characters. It is enough to represent characters in any language, even Chinese and hieroglyphics.

A problem arises when the computer retrieves a piece of data from its memory. Imagine that the data is 01000001. Is this the number 65, or is it A?
They are both stored in the same way, so how can it tell the difference?
The answer is that characters and numbers are stored in different memory locations so it knows which one it is by knowing whereabouts it was stored.

Number Systems

Number System defines a set of values used to represent 'quantity'. The term Base **(or Radix)** refers to the total number of digits available in a number system.

5. Distinguish Arithmetic Errors

Overflows

- For an unsigned number, overflow happens when the last carry (1) cannot be accommodated
- For a signed number, overflow happens when the most significant bit is not the same as every bit to its left
- when the sum of two positive numbers is a negative result
- when the sum of two negative numbers is a positive result
- The sum of a positive and negative number will never overflow

Range

The formula used for calculating the range of values that can be represented is

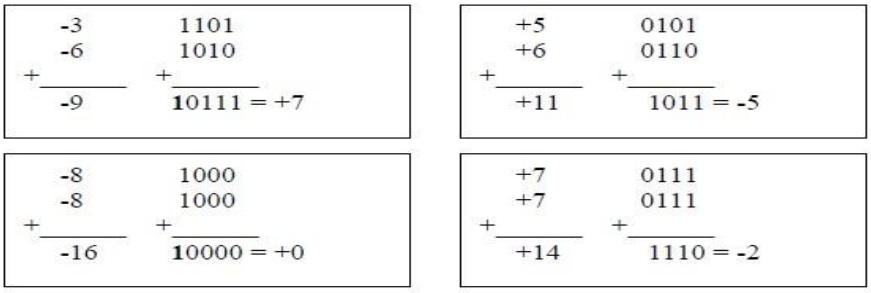
- (2^{n-1}) to +(2^{n-1}-1)

Range = **Lowest Negative** =-8; **Highest positive** = +7

- Therefore range is from 1000 to 0111, it -8 to +7
- If number is added and the result is more than +7, there is an **overflow**.
- If the result is less than -8, it is an **underflow.**
- If the sign changes after addition, it's an **overflow.**
- In general, *overflow* occurs if both numbers to be added have the same sign, otherwise no overflow occurs.

- One way to detect overflow is to check the sign bit of the sum. If the sign bit of the sum does not match the sign bit of **x** and **y**, then there's overflow. This only makes sense.
- Suppose **x** and **y** both have sign bits with value 1. That means both representations represent negative numbers. If the sum has sign bit 0, then the result of adding two negative numbers has resulted in a non-negative result, which is clearly wrong. **Overflow** has occurred.
- Suppose **x** and **y** both have sign bits with value 0. That means, both representations represent non-negative numbers. If the sum has sign bit 1, then the result of adding two non-negative numbers has resulted in a negative result, which is clearly wrong. **Overflow** has occurred.

This suggests that one way to detect *overflow* is to look at the sign bits of the two most significant bits and compare it to the sum. Refer to diagrams below:



6. Interpret Arithmetic Errors

Range:

Allowable values for a register representation of data, starting from the lowest (minimum) allowable to the highest (maximum) allowable values. It can also be defined as the difference between the lowest and the highest acceptable values.

Accuracy and Errors

Floating and fixed point numbers will be accurate to the smallest number they can represent.

Accuracy: is the degree of closeness (nearness) of measurements of a quantity to that quantity's actual (true) value.

Precision: **precision** is the degree to which further measurements or calculations show the same or similar result. More bits for precision is a measure of reliability (repeatedly give the same value). More bits for mantissa means the number will be more precise, i.e by having more significant figures

To **increase accuracy**, either you can:

- Increase the number of bits used for the mantissa
- Then reduce the number of bits for the exponent

However, the range of numbers represented is reduced because the size of the index of the power of two is reduced

Round-Off Errors

Rounding: Expressing a number to the nearest whole/decimal/binary number. For example 3.9569 rounded to 3 decimal places is 3.957. If rounded to the nearest whole number, it becomes 4.

Often we cannot represent a denary fraction exactly even if we allow many bits in memory. Therefore the number stored is "rounded off" to the closest possible binary equivalent.

In rounding, the least significant bit may be increased depending on digits removed. The result should represent the value that is nearest to the original value, e.g

100.1 = 100 (2 s. f)
10101 = 1100 (2 .f)
11011 = 11000 (2 s.f)
11.101 = 100 (2 s.f)

1100 =1100 (3 s.f)
1101 = 1110 (3 s.f)
111.1101 = 1110.11 (6 s.f)
1110.1110 = 1111.00 (6 s.f)

Truncation Errors

Truncation is shortening by cutting off some characters/ending text abruptly at a certain point; e.g. 3.9569 truncated to 3 decimal places is 3.956. If truncated to the nearest whole number, it becomes 3.

Truncation error is difference between a truncated value and the actual value. A truncated quantity is represented by a numeral with a fixed number of allowed digits, with any excess digits "chopped off" Often, in either floating or fixed point systems, results are calculated with too many places of accuracy to be represented. We get this type of error when trailing bits are truncated to fit the result in the memory location available.

Truncation works as follows:

100.1 = 100 (2 s. f)
10101 = 10000 (2 .f)
11011 = 11000 (2 s.f)
11.101 = 11 (2 s.f)

Overflow

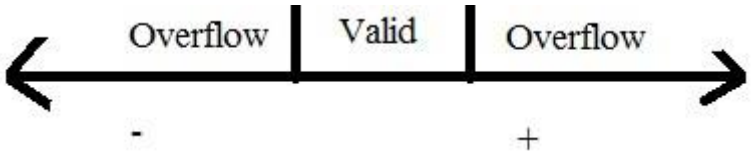
- It is a computational process which produces a result so large that it cannot be represented.
- Overflow occurs when
  - a number is divided by a small number or
  - When two large numbers are multiplied together.

Underflow

- An underflow is produced when a result that is smaller in magnitude than the smallest number that can be represented.
- It occurs when a small number is divided by a large number or
- When small numbers are multiplied together.

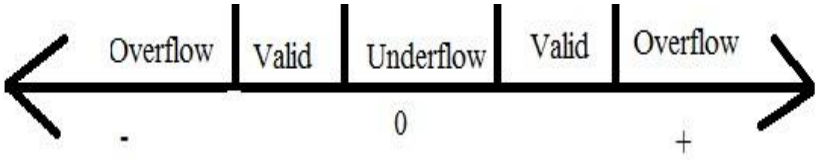
Overflow and underflow representation

(a) Integer representation



If the result is outside the possible range, then it is **overflow**.

(b) Floating Point representation



There exists the largest and smallest positive value. Around 0, there exists a range of values that cannot be represented (stored) and this is **underflow**.

NB: Overflow and underflow occur when a result of calculations falls outside the range of values permitted by the representation of the number.

Questions

(a) Using the binary number 11100111111101 as an example, show how to use the binary representation of a number to work out its value in hexadecimal with minimum amount of calculation. [3]
(b) A floating point number system uses 8-bit numbers, 5 bits for the mantissa and 3 bits for exponent. Convert the following binary number to denary 01101010 [2]

(c) Using an 8-bit byte for the mantissa and an 8- bit byte for the exponent, show –15½ as a 2 byte, normalised, floating point number. [4]
Explain the
(a) character set, [2]
(b) ASCII, [2]
(c) EBCDIC. [2]

qxn 2 Zimsec Specimen paper 2018

1 (a) Express the denary number 78 as
(i) a binary number stored in an 8 bit byte,
(ii) a hexadecimal number,
(iii) a number stored in binary coded decimal (BCD). [6]
(b) Explain how the binary value of 78 can be used to write down the equivalent octal value with a minimum amount of calculation [3]
(c) (i) Convert -63 and -94 into 2's complement, 8 bit, binary numbers. [2]
(ii) Add the binary values obtained in (i) together. [2]
(iii) Comment on the result that you obtained in (ii). [2]

(a) (i) 01001110 (1 per nibble) (2)
(ii) 4E (1 per digit) (2)
(iii) 01111000 (1 per nibble) (2)
(b) -Bits arranged in threes from the right
-Need to add leading zero
- 001 001 110
-Each group of three bits converted (to denary/octal)
(1 per -, max3) (3)
(c) (i) 11000001
10100010 (2)
(ii) 1,01100011
(1 for answer, 1 for indication of overflow, allow ft) (2)
(iii) -Overflow
-Answer is positive…
-because of overflow from +ve bits into –ve bit
-Processor recognizes error because carry in to MSB is different from carry out.
(1 per-, max 2) (2)

2 (a) (i) Express the number 93 as an 8 bit binary number. [2]
(ii) Express the number 93 as a number in octal. [2]
(iii) Express the number 93 as a number in hexadecimal. [2]
(b) (i) Explain how to use the binary representation of a number to work out its value in octal. [2]
(ii) Describe the connection between binary representation and hexadecimal. [2]

(a) (i) 01011101
(1 per nibble) [2]
(ii) 135
(1 for 1, 1 for 35) [2]
(iii) 5D
(1 per digit) [2]

(b) (i) -Group the bits in threes
-from the LSB
-Change the binary groups to denary
(1 per -, max 2) [2]
(ii) -Groups of 4 bits
-Give hexadecimal values [2]

3. (a) Show how the denary number –90 can be represented, using 8 bits, in:
(i) sign and magnitude,
(ii) two's complement. [2]
(b) The denary number 10¾ is to be represented as a floating point binary number using 12 bits. The first 8 bits are to be used for the mantissa and the remaining four bits are to be used for the exponent.
(i) Explain what is meant by the mantissa of a floating point number. [2]
(ii) Explain what is meant by the exponent of a floating point number. [2]
(iii) Show why 001010110101 is a floating point representation of 10¾. [3]
(iv) Normalise the floating point value given in (iii). [2]

(a) (i) 11011010
(ii) 10100110
(1 per dotty) [2]
(b) (i) -The fractional part of the representation
-Place value of MSB is –1…
-remainder of bits are ½, ¼…
-Holds the magnitude of the data.
(1 per -, max 2) [2]
(ii) -Is a two's complement integer which…
-holds the power of 2…
-by which the mantissa must be multiplied...
-to give the original value.
(1 per -, max 2) [2]
*(iii) 0.0101011 * 10 ^ 0101*
= 1010.11
= 8 + 2 + ½ + ¼
Alternative:
10 = 1010 and .75 = .11
10.75 = 00101011 x 10^101
Point moves 5 places
(1 per line, max 3) [3]
(iv) 01010110 0100
(1 for mantissa, 1 for exponent) [2]

4. Using an 8 bit byte for the mantissa (fraction) and another 8 bit byte for the exponent (characteristic)
(a) show
(i) 10½
(ii) -10½
as 2 byte, normalised, floating point numbers. [4]

(b) Show the bit pattern that represents
(i) the largest positive
(ii) the smallest magnitude negative
number that can be represented using this 2 byte normalised floating point form. [4]

(a) (i) 01010100 / 00000100
(ii) 10101100 / 00000100
(1 per byte) (4)

(b) (i) 01111111 / 01111111
(ii) 10111111 / 10000000
(1 per byte) (4)

5. (a) Express the denary value 109 as
(i) a binary number using an 8-bit byte;
(ii) an octal number;
(iii) a hexadecimal number. [6]

(b) Numbers are held in floating point form with one byte for the mantissa (fraction) and one byte for the exponent (characteristic). All values are held in two's complement form and the mantissa is normalised.
Using this format, write down the binary floating point values and the denary values of
(i) the largest magnitude, positive number;
(ii) the smallest magnitude, positive number;
(iii) the largest magnitude, negative number;
(iv) the smallest magnitude, negative number.
(The denary values may be left as a product of a power of 2). [8]

(c) Explain how accuracy can be improved in a floating point representation and state an effect it can have on the number represented. [3]

5(a) (i) 01101101 (1 for binary, 1 for 8 bits)
(ii) 155 (1 for 1, 1 for 55)
(iii) 6D (1 for 6, 1 for D) (6)
(b) (i) 01111111/01111111 = 127/128x2^127
01000000/10000000 = 1/2x2^-128 (2^-129)
10000000/01111111 = -1x2^127
10111111/10000000 = -65/128x2^-128 (8)

(c) Increase the number of bits used for the mantissa
by reducing the number of bits for the exponent
The range of numbers is reduced because
The size of the index of the power of two is reduced
(1 per point, max 3) (3)

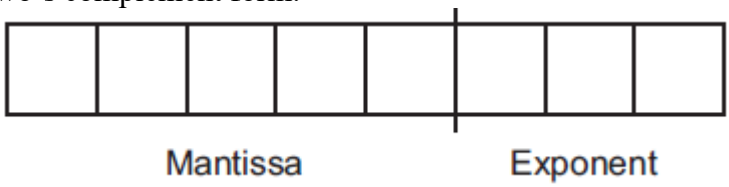
6.(a) Express the decimal number 109 as
(i) a binary number stored in an 8 bit byte; [2]
(ii) a number in binary coded decimal (BCD); [2]
(iii) a hexadecimal number. [2]

(b) A particular computer stores numbers as 8 bit, two's complement, binary numbers. 01011101 and 11010010 are two numbers stored in the computer.
(i) Write down the decimal equivalent of 11010010. [2]
(ii) Add the two binary values together and comment on your answer. [3]

(a) (i) 01101101 (1 per nibble) (2)
(ii) 0001 0000 1001 (1 for use of 12 bits, 1 for correct answer) (2)
(iii) 6D (1 per digit) (2)
(b) (i) -46 (1 for negative, 1 for 46) (2)
(ii) -(1)00101111/result = +47
-a positive and negative have been added together and the result is positive

-because the larger value was positive.
-there was carry in and out of MSB therefore ignore carry out, (result is correct).
(1 per -, max 1 for either answer, max 2 for discussion, max 3) (3)

7. A computer stores fractional numbers in floating point binary representation. Five bits are used for the mantissa and three bits for the exponent. All values are stored in two's complement form.



(a) By using a diagram of this representation, state the value of each of the bits. [4]
(b) By using 2 ½ as an example, explain how real numbers can be shown in normalised form in this representation. [3]
(c) State the floating point binary value of - ¾ in this representation. [2]

(a) Bit headings: –1 (1 mark)
½, ¼, 1/8, 1/16 (1 mark)
–4 (1 mark)
2, 1 (1 mark) [4]
(b) 2 ½ = 10.1
= 0.101 x 2 ^ 10
= 01010010 (On its own, worth 2 marks)
Normalised because first 2 digits are different.
(1 per line, max 3) [3]
(c) 10100000
(1 for first 5 bits, 1 for last 3 bits) [2]

8. (a) Express the denary number 94 as:
(i) a BCD value, [2]
(ii) an octal value, [2]
(iii) a hexadecimal value. [2]
(b) (i) Work out the answer to the following binary addition sum. (All values are given in two's complement form. You should show your working.)

01001101
00101011
01000101 +
_____ [2]

(ii) Explain why the binary result does not give the correct answer. [1]

(a) (i) 10010100 (1 per nibble) [2]
(ii) 136 (1 for 1, 1 for 36) [2]
(iii) 5E (1 per digit) [2]
(b) (i) 10111101 (1 for answer/1 for carries) [2]
1 1111
(ii) Negative result when originals all positive/carrying in to MSB but not out/overflow. [1]

10. A computer stores floating point numbers using 2 eight-bit bytes.The first byte is used to store the mantissa and the second stores the exponent.
(a) The normalised form of the floating point representation of 9 ½ is
0100110000000100.
(i) Explain this representation of 9 ½ . [4]
(ii) Give the floating point value of 22 ¼
using this representation. [2]

(b) It is decided to change the representation by using 10 bits for the mantissa and 6 bits for the exponent.
Explain the effect of this decision on the range and accuracy of the data represented. [4]

(a) (i) Mantissa is 01001100
-Created by 9 ½ = 1001.1
-Point moved to be in front of first 1 and 0 placed in front
Exponent is 00000100
-created by number of places point is moved
-4 = 1002
(1 per -, max 4) [4]

(ii) -Mantissa is 01011001
-Exponent is 00000101 [2]
(b) -Range is decreased...
-because power of two to multiply mantissa by is decreased
-Accuracy is increased...
-because more digits/bits (are represented after the binary point). [4]

11. Floating-point numbers in a particular computer system are stored using 12 bits. The first 6 bits are used for the storage of the mantissa and the second set of 6 bits is used to store the exponent.
(a) One way to represent 6.5 as a floating-point number in this representation is
001101000100
Explain why this representation is equivalent to 6.5 [4]
(b) (i) Using the representation above to help you, write the number 6.5 as a floating-point number in normalised form. [2]
(ii) Explain the effects of changing the representation so that 8 bits are used for the mantissa and 4 bits for the exponent. [2]
(c) The numbers 011011001101 and 101100001110 are stored with 6 bits for the mantissa and 6 for the exponent.
Add the exponents of the two floating-point numbers together. [2]

(a) - Mantissa is 001101 and exponent is 000100
- 001101=1/4+1/8+1/32 = 13/32
- Exponent = 4
*- Number represented = 13/32 * 2^4*
*- = 13/32 * 16*
= 6.5
Alternative:
- Mantissa is 001101 and exponent is 000100
- Mantissa is 0.01101

- Exponent is 000100 = 4
- Therefore number is 110.1
= 6.5
(1 per -, max 4) [4]
(b) (i) 011010 000011
(1mark for mantissa, 1 mark for exponent) [2]
(ii) -Accuracy of representation is increased
-Range is decreased [2]
(c) 011011 [2]

12. (a) Represent
(i) +102,
(ii) +117
as 8-bit numbers in two's complement form. [2]
(b) (i) Add the answers in part (a) together to give a binary result. [2]
(ii) Turn your binary answer into an equivalent denary result. [2]
(iii) Explain the validity, or otherwise, of your result. [2]

(a) (i) 01100110
(ii) 01110101 (2)
(b) (i) 11011011 (1 per nibble) (2)
(ii) -128 + 91 = -37 (2)
(iii) The original numbers are positive
The answer is negative
There has been an overflow from the positive part of the byte to the negative.
(1 per point, max 2) (2)

Note: Follow through in part (b) on wrong answers in part (a) of 10011010 and 10001011
They give the answer 00100101 (for 2 marks)
Which gives the answer 37 (for 2 marks)
Which gives: Originals are negative
Answer is positive
Overflow out of byte (any two of the three for final 2 marks)

13 (a) (i) Explain what is meant by the character set of a computer.
(ii) Explain how a character is represented in a computer. [4]
(b) Explain the representation of integers in a computer. [3]
(c) Explain how a Boolean value is stored in a computer. [2]

13 (a) (i) – The symbols recognised/used by the computer
– Often equates to the symbols on the keyboard
(ii) – Represented by a set of bits...
– Unique to that character
– The number of bits needed is equal to 1 byte / 2 bytes
– ASCII/Unicode is a common set
(1 per –, max 3 per dotty, max 4) [4]
(b) – Bits are used to store the correct binary representation of the integer
– Leading zeroes included to complete required number of bits
– Standard number of bits irrespective of size of integer
– Concept of short and long integer dependent on sizes of integers
– Two's complement used to represent negative numbers
(1 per –, max 3) [3]
(c) – As a single bit/byte
– a 0 or a 1/a byte of all 0s or all 1s [2]
Do not accept Y/N or True/False

14(a) (i) Explain what is meant by the character set of a computer. [1]
(ii) Describe how the character set is represented in the computer system. [2]

14(a) (i) Those symbols that the computer (software) can recognise (1)
(ii) - As binary codes
- ASCII/EBCDIC
- Using 7,8,15,16 bits
- The number of bits = 1 byte
(1 per point, max 2)

Networking

1. Compare OSI and TCP/IP models

Transmission protocols

A protocol is a set of rules that govern how data is transferred in a network. It defines the rules on how network devices communicate, e.g the TCP/IP. This includes:

- how to interpret signals
- how to identify 'oneself' and other computers on a network
- how to initiate and end networked communications

1. OSI (Open Systems Interconnection)

This is a model of communication designed by the International Standards organization (ISO). The OSI model allows computers from different manufacturers or origin to be connected together. The idea is that suppliers must produce hardware or software to implement any of the seven layers while other suppliers provide those of other layers. This promotes standardization. The seven layers are:

- **Application Layer** – defines how user applications interface or access communication services, initiates or accepts a request, provide network applications like data transfer, messaging, operating system functions, etc
- **Presentation Layer** – deals with how information is presented, e.g ASCII, adds formatting or data transformation,(e.g. from ASCII to Unicode), data compression and data encryption.

  Session Layer

  – creates and terminates sessions, e.g. login session, file transfer session, adds traffic-flow control information, etc.
  - Session layer is responsible for deciding on the communication method
- **Transport Layer**

  - allows error correction during transmission, maintains flow control, allows data recovery, and allows routing, addressing and multiplexing of signals.

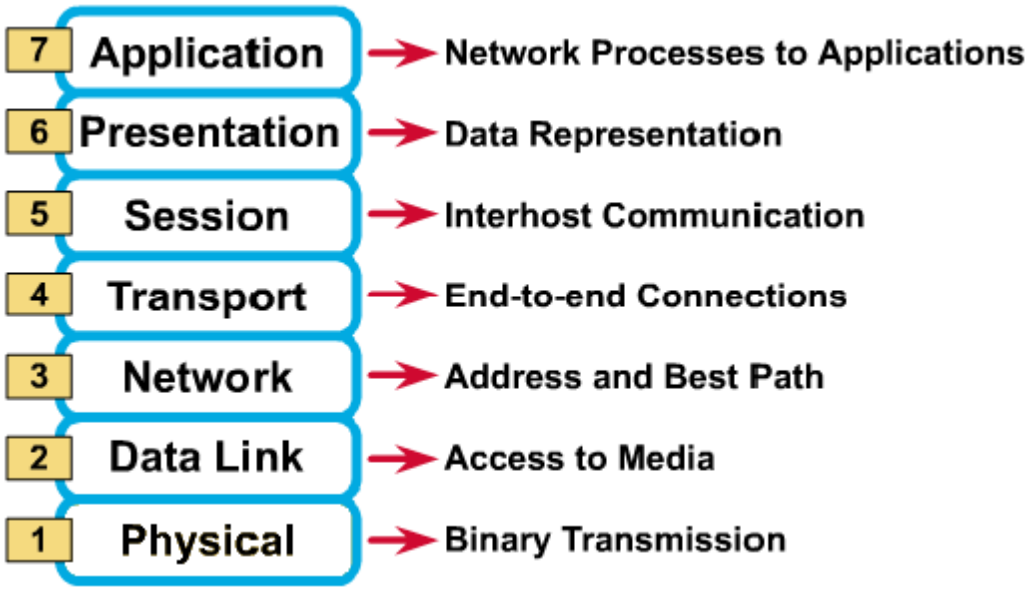
  -Transport layer implements communication methods decided on session layer

  - Implements the functions necessary to send data to the communication partner.
  - These mechanisms include multiplexing data from different applications, establishing data integrity, and management of virtual circuits.
    - ***Multiplexing*** - method of combining data from upper layers and sending them through the same data stream allowing more than one application to communicate with the communication partner at the same time.
    - ***Virtual circuits*** are the methods of setting up a communication path to the receiver which may physically change depending on the network, but the path remains open through a virtual link. The *Transport layer is responsible for establishing, maintaining, and disconnecting* the virtual circuits.
    - ***Data integrity***

      Use buffering, source quench, and windowing to ensure integrity.
      - *Buffering* is maintained on the receiving computer. As data flows in faster than can be processed, some data is placed in a buffer and held until the computer has the time to process it.

- *Source quench* is a technique where the receiving computer can send a control message back to the sending computer when too much data is being received. The sending computer then will delay sending any more data until the receiving computer can finish processing the current data.
- *Windowing*, works on the principle that the receiver tells the sender how much data it can send at one time (window size).

- **Network Layer** - adds sequencing and address information, sets logical protocols, creating frames (consisting of address fields, control filed, date, and error control field)
- **Data Link Layer** - provides error-checking and formats data for physical transmission, defined network type and packet sequencing, used for synchronisation.
- **Physical Layer**
  -It is responsible for sending data and receiving data across a physical medium. The data is sent in bits, either a 0 or a 1.
  – level of actual hardware, define physical characteristics of network, defines the Data Terminal Equipment (DTE) and the Data Circuit-Terminating Equipment (DCE)



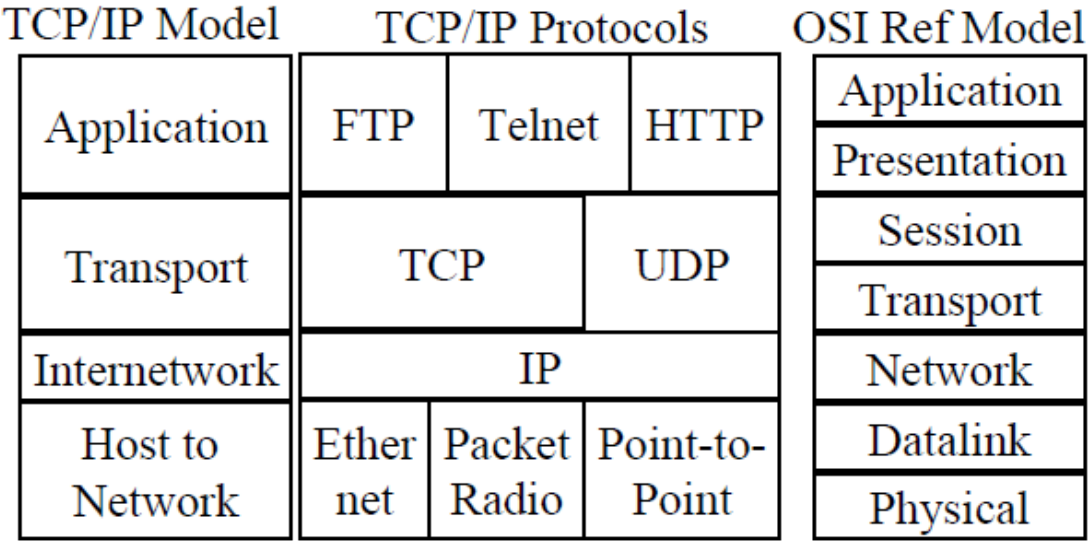
At each level, additional information is added to allow service to be provided. This layered model is also called **protocol stack**

2. TCP/IP (Transmission Control Protocol/Internet Protocol)

- protocol governing the transmission of data
- data is divided into packets to which addressing information, error correction code and identification are added
- the packets travel to their destination over the network and the receiving PC checks for mistakes and pieces the data together in the right order
- **TCP/IP (Transmission Control Protocol Internet Protocol)**
  TCP: It ensures that data is transmitted accurately
  IP: It ensures that data is transmitted to its correct destination (IP address). Every device on the internet has its IP address. It also ensures that packets are rearranged to the original message on arrival of their destination.

TCP/IP Reference Model

- TCP = Transport Control Protocol
- IP = Internet Protocol (Routing)



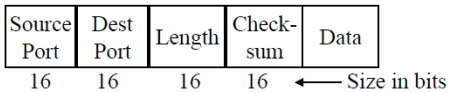
A network communication protocol: a standard method for transmitting data from one computer to another across a network. Some of the protocols are:

i. **HTTP (HyperText Transfer Protocol)**
This is a protocol that defines the process of identifying, requesting and transferring multimedia web pages over the internet. It is used for transferring data across the internet, usually between servers and computers on the internet. It is based on the client –server relationship. It uses TCP/IP to transmit data and messages

ii. **FTP (File Transfer Protocol)** it is a protocol used to transfer data from one computer to another. It is often used to download software from the internet, and it uses the TCP/IP protocol in doing this. However, FTP has no security to data as the data is not encrypted prior to its transmission.

iii. **TELNET**
This is a network protocol that allows a computer user to gain access to another computer and use its software and data, usually on a LAN and on the Internet. It allows users to access data stored on servers from their terminals. Telnet allows computers to connect to each other and allows sharing of data and files. Telnet has security problems especially on the internet.

iv. **User Datagram Protocol (UDP)**
- **Provides** end-to-end service
- UDP is connectionless and simple.
- Provides No flow/error control.
- Provides port addressing
- Error detection (Checksum) is optional.



v. **VoIP (Voice Over Internet Protocol)**
It is a method of using the internet to make ordinary voice telephone calls. Thus it is a way of having phone conversations using the internet as a way of communication. By VoIP, international and long distance calls are of the same price as local calls and sometimes are for free. However, the system does not offer emergency calls. An example of VoIP is Skype.

Comparison of OSI and TCP/IP

OSI	TCP/IP
Introduces concepts of: 1. Services 2. Interfaces 3. Protocols	No explicit definitions of service, interface and protocol
• Is more general than TCP/IP • Model built before protocols	• Model describes protocols • Suitable only for TCP/IP networks
Connection-oriented transport l. Network layer – both c/o and c/l	Transport – both c/o and c/l Network layer - connectionless

2. Explain the format of an IP address

IP addressing is a hardware-independent convention which in principle allows every computer attached to the Internet to be given a unique *logical* address

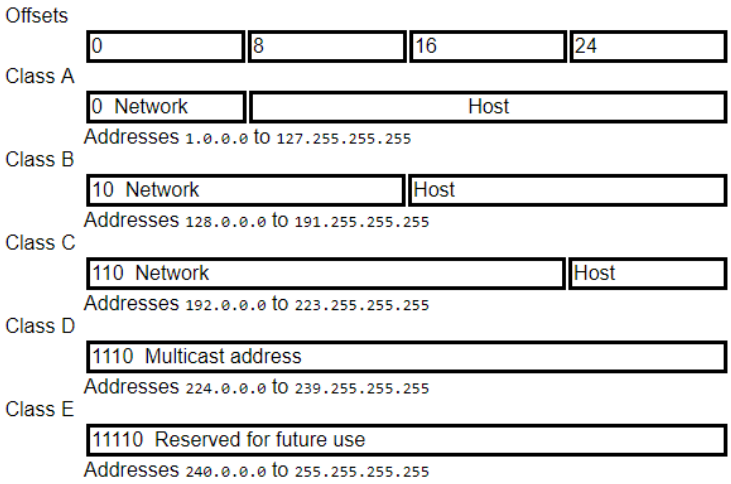
IP address

- an identifier for a computer or device on a TCP/IP network.
- TCP/IP protocol networks route messages based on the IP address of the destination.

IP address Format

- The format of an IP address is a 32-bit numeric address written as four numbers separated by periods with each number ranging from zero to 255 eg, **1.160.10.240** could be an IP address.
- The four numbers in an IP address are used in different ways to identify a particular *network* and a *host* on that network.

Originally IP addresses were divided into five *classes* as shown below. Classes A, B and C are the most important: the initial bits determine which class an address belongs to, and the classes differ in how much of the address is taken up with the *network address* and how much with the *host address*.



Class A - supports 16 million hosts on each of 126 networks

Class B - supports 65,000 hosts on each of 16,000 networks

Class C - supports 254 hosts on each of 2 million networks

- The number of unassigned Internet addresses is running out, therefore the system based on classes A, B, and C is gradually being replaced by adoption of IPv6.
- In **IPv6** the IP address size is increased from 32 bits to 128 bits.

Static and Dynamic IP Addresses

An IP address can be *static* or *dynamic*.

- A ***static*** IP address will never change and it is a permanent Internet address.
- A ***dynamic*** IP address is a temporary address that is assigned each time a computer or device accesses the Internet.

3. Distinguish between public and private IP addresses

Public (external) IP addresses

- A public (or external) IP address is the one issued by ISP (Internet Service Provider) to identify a network to the outside world.
- It is an IP address that is unique throughout the entire Internet.

Private (internal) IP addresses

- **Private IP address** is an **IP address** that is reserved for internal use behind a router or other Network **Address** Translation (NAT) device, apart from the **public**.
- Router issues private (or internal) IP addresses to each network device inside the network.
- This provides unique identification for devices that are within a home network, such as computer, tablet, etc.

**The router controls all the network traffic, both within a home network and outside of it, to the Internet. It is the router's job to make sure that data flows to and from all the correct places.*

4. Explain the role of DNS

Domain names are alphanumeric names for IP addresses e.g., neon.cs.virginia.edu, www.google.com, ietf.org

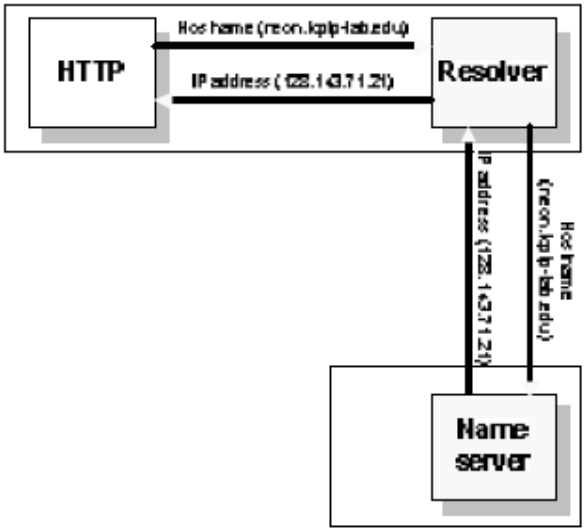
DNS

• is an Internet-wide distributed database that translates between domain names and IP addresses

- A system which permits humans to use names and machines to use addresses
- a distributed database where data is maintained locally, but available globally
  - DNS uses
    - replication to achieve robustness
    - caching to achieve adequate performance
  - DNS is composed of *namespace*, *name servers* and *resolvers*
    - a ***namespace*** - the database's structure ie is a hierarchical and logical tree structure for naming system
    - ***name servers*** - store data from specific segments of the database, Answer questions from resolvers
    - ***resolvers*** - translate applications' requests for data into DNS queries
    - Interpret name server's responses

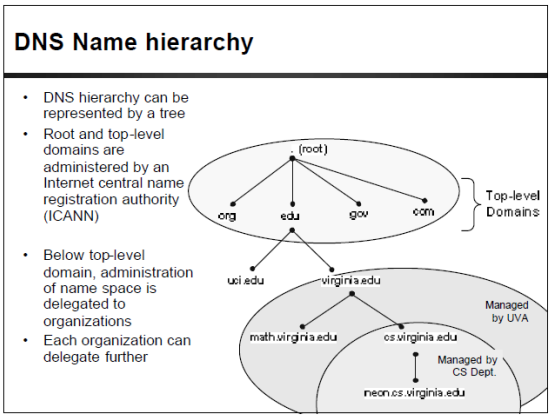
Resolver and name server

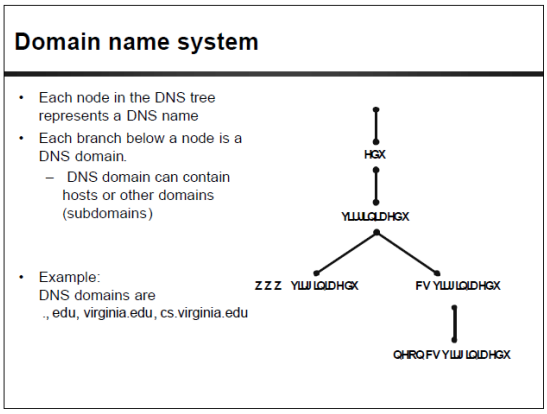
1. An application program on a host accesses the domain system through a DNS client, called the **resolver**
2. Resolver contacts DNS server, called name server
3. DNS server returns IP address to resolver which passes the IP address to application



- Reverse lookups are also possible, i.e., find the hostname given an IP address

maregat@mbizohighschool.ac.zw



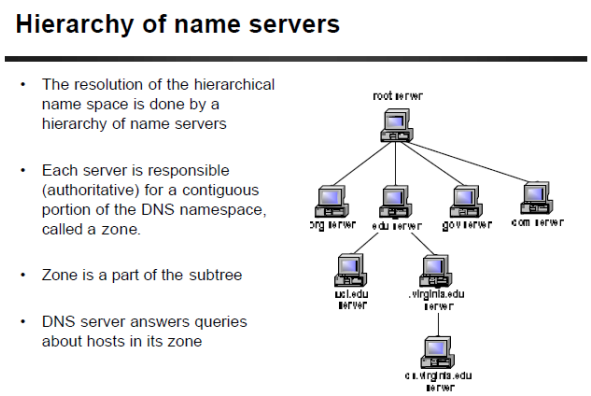


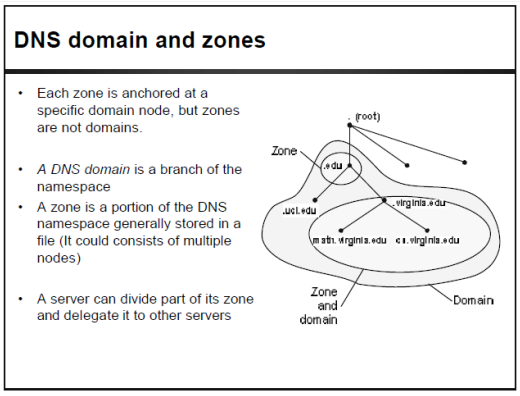
Domain names

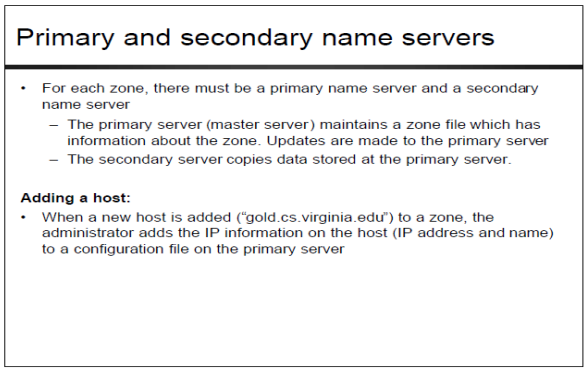
• Hosts and DNS domains are named based on their position in the domain tree
• Every node in the DNS domain tree can be identified by a unique Fully Qualified Domain Name (FQDN). The FQDN gives the position in the DNS tree.

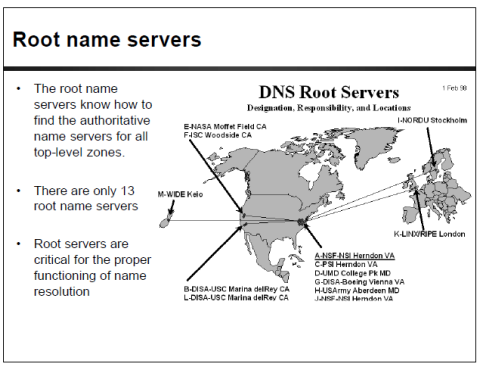
cs.virginia.edu or cs.virginia.edu.

• A FQDN consists of labels (“cs”,“virginia”,”edu”) separated by a period (“.”)
• There can be a period (“.”) at the end.
• Each label can be up to 63 characters long
• FQDN contains characters, numerals, and dash character (“-”)
• FQDNs are not case-sensitive









5. Describe Routing Information Protocol(RIP) and Open Shortest Path First (OSPF) routing protocols

Routing Information Protocol (RIP)

- is an interior gateway protocol that routers can use to exchange network topology information.
- Typically used in small to medium-sized networks.
- uses distance vector i.e shares a list of distance-vectors with each of its neighbours periodically
- A router sends the contents of its routing table to each of its adjacent routers every 30 seconds.
- Each router computes new distances and replaces entries with new lower hop counts
- When a route is removed from the routing table, it is flagged as unusable by the receiving routers after 180 seconds, and removed from their tables after an additional 120 seconds.

Open Shortest Path First (OSPF)

- An Interior Gateway Protocol (IGP) for the Internet, used to distribute IP routing information throughout a single Autonomous System (AS) in an IP network.
- A link-state routing protocol, in which the routers exchange topology information with their nearest neighbors.
- The topology information is flooded throughout the AS, so that every router within the AS has a complete picture of the topology of the AS.
- the complete knowledge of topology allows routers to calculate routes that satisfy particular criteria.
- This picture is then used to calculate end-to-end paths through the AS, normally using the Dijkstra algorithm.
- the next hop address to which data is forwarded is determined by choosing the best end-to-end path to the eventual destination.
- It provides support for multiple paths of equal cost.
- It provides a multi-level hierarchy (two-level for OSPF) called "area routing," so that information about the topology within a defined area of the AS is hidden from routers outside this area which increase the level of routing protection and a reduce routing protocol traffic.
- All protocol exchanges can be authenticated so that only trusted routers can join in the routing exchanges for the AS.

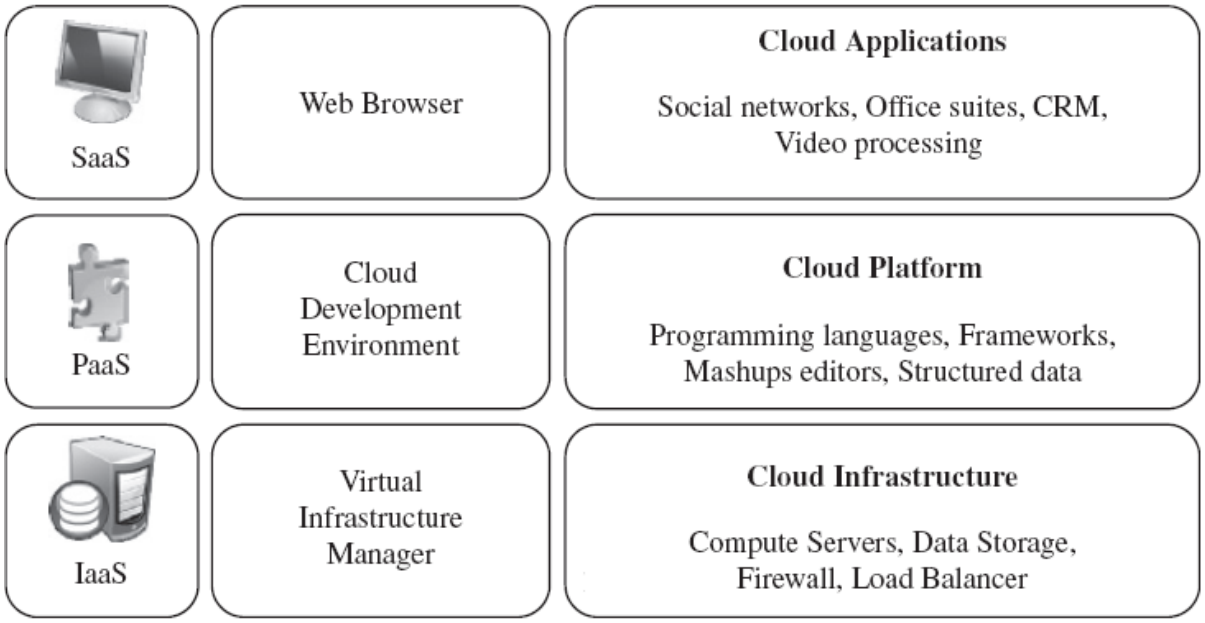
RIP	OSPF
- Uses distance vector algorithm (Bellman-Ford) - Shares knowledge about the entire Autonomous System - Shares only with neighbors - Shares at regular intervals	- Uses true metrics (not just hop count) - Allows load balancing across equal-cost paths - Authenticates route exchanges - Quick convergence - Large networks are subdivided into a backbone network and areas - Each area has multiple subnets. Each subnet has a designated router. - *Link state* routing ⇒ Each router broadcasts its connectivity with neighbours to entire area

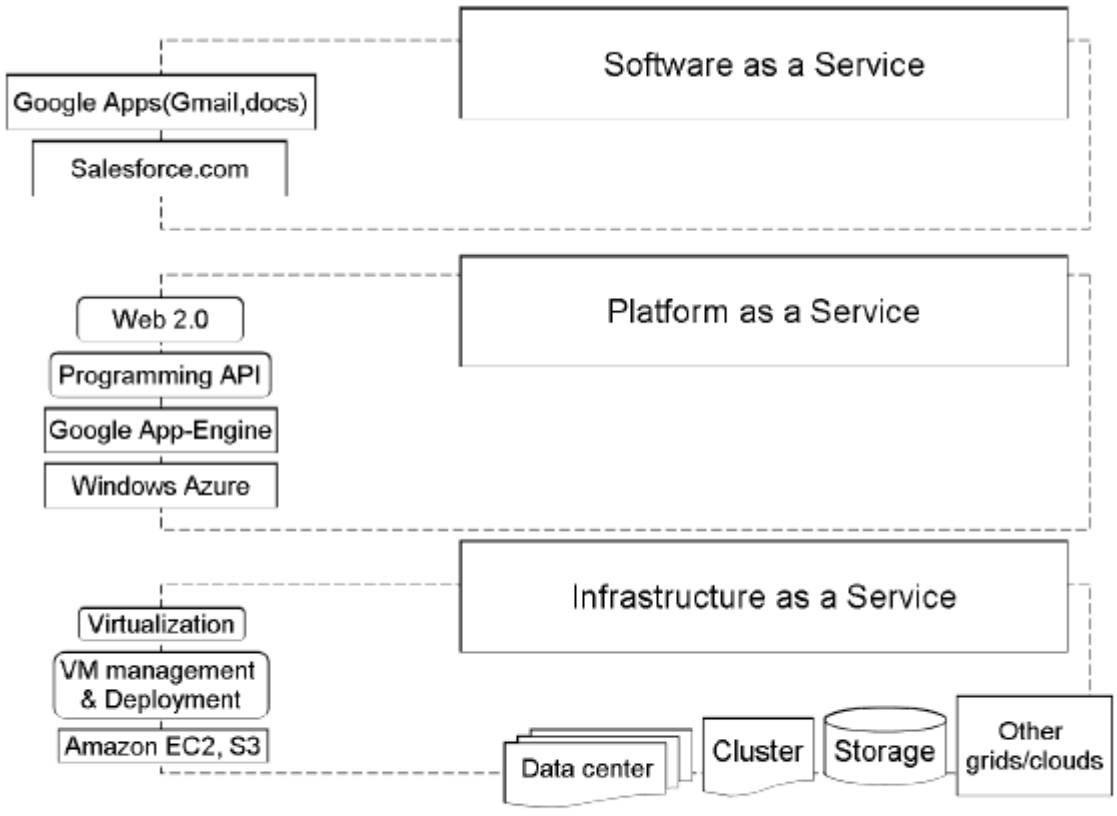
6. Differentiate cloud service models

Cloud Computing

- cloud computing is a distributed computing environment that enables the users to access and exchange their resources (applications and data) remotely and provides services to use the remote hardware and software within a network without the knowledge of technological infrastructure
- Cloud computing is a paradigm of distributed computing to provide the customers on-demand, utility based computing services.
- Cloud itself consists of physical machines in the data centres of cloud providers.
- Different cloud provider provides cloud services of different abstraction level. E.g. Amazon EC2 enables the users to handle very low level details where Google App-Engine provides a development platform for the developers to develop their applications.

Cloud computing is able to provide a variety of services at the moment but main three services are Infrastructure-As-A-Service, Platform-As-A-Service and Software-As-A-Service also called as service model of Cloud computing





Cloud Service stack

Software as a service (**SaaS**)

- is a **software** distribution model in which a third-party provider hosts applications and makes them available to customers over the Internet.
- A service provider hosts the application at its data center and a customer accesses it via a standard web browser.

There are a few major characteristics that apply to most SaaS vendors:

- Updates are applied automatically without customer intervention
- The service is purchased on a subscription basis
- No hardware is required to be installed by the customer

SaaS is also known as hosted software or on-demand software.

Platform as a Service (**PaaS**) or **application platform as a Service** (**aPaaS**) or **platform base service**

- Provides a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure necessary for development and launching of an application.
- Is whereby a third-party provider delivers hardware and software tools required for application development to users over the internet.
- PaaS frees users from having to install hardware and software to develop or run a new application.

Infrastructure as a service (IaaS) or **hardware as a service (HaaS).**

- a service model that delivers computer infrastructure on an outsourced basis to support enterprise operations.
- provides virtualized computing resources (*i.e provides hardware, storage, servers and data center space or network components; and may also include software.*)over the internet
  - IaaS provider provides policy-based services and is responsible for housing, operating and maintaining the equipment it provides for a client

PaaS vs. SaaS vs. IaaS

- With ***IaaS***, a provider supplies the basic compute, storage and networking infrastructure along with the hypervisor (the virtualization layer). Users must then create virtual machines, install operating systems, support applications and data, and handle all of the configuration and management associated with those tasks.

- With ***PaaS***, a provider offers more of the application stack than IaaS providers, adding operating systems, middleware (such as databases) and other runtimes into the cloud environment.

- With ***SaaS***, a provider offers an entire application stack. Users simply log in and use the application that runs completely on the provider's infrastructure.

Attribute System	Features	Challenges
Infrastructure-as-a-Service	1. Elasticity 2. Transferring the risks 3. Reduced operational costs 4. Availability of latest infrastructure 5. Inter-operatability 6. Disaster recovery	1. Temperature of cloud places need to be maintained 2. System should be power failure tolerant 3. Selection of infrastructure hardware is very important 4. Connection between cloud and hardware should be a high bandwidth channel 5. Storage of cloud should be able to fulfill the changing demands of large data size 6. Loss of control
Software-as-a-Service	1. Cost minimization 2. High throughput 3. Time saving 4. Availability of high tech services 5. High availability 6. Reduced administration cost	1. Data security is highly preferred feature 2. High availability requirement 3. Authentication and authorization 4. Data integrity 5. Data privacy 6. Network security 7. Cloud standardization 8. Deployment of cloud resources in different countries results in conflict of rules 9. Data backup 10. Web application security 11. Data confidentiality 12. Virtualization
Platform-as-a-Structure	1. Access of high level infrastructure 2. Flexibility 3. Ready to use services 4. Scalability 5. Less administration cost	1. Limited APIs 2. Data Lock-in 3. Auditability 4. Performance is unpredictable 5. Lack of control over low level security 6. Data inaccessibility between applications 7. Vulnerabilities of web applications and SOA

7.Describe Cloud Types

Cloud Deployment Models/Types

There are three commonly-used cloud deployment models: *private, public, and hybrid* with an additional model the *community* cloud, which is less-commonly used.

A ***private*** cloud is built and managed within a single organization using such software as VMWare, vCloud Director, or OpenStack.

- Used solely for the organization's internal purpose therefore security and network bandwidth are not critical issues for private cloud.

A ***public*** cloud is a set of computing resources provided by third-party organizations. eg Amazon Web Services, Google AppEngine, and Microsoft Azure.

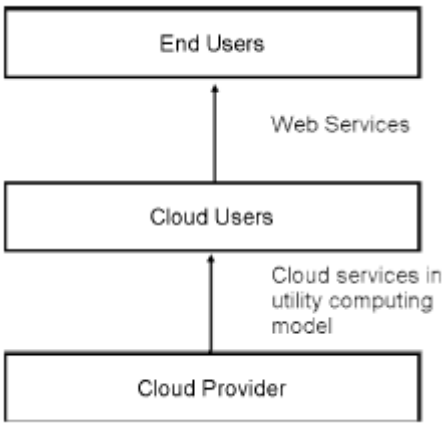
- An organization rents cloud services from cloud providers on demand basis.

A ***hybrid*** cloud is a mix of computing resources provided by both private and public clouds.

- is composed of multiple internal or external clouds, a scenario like when an organization moves to public cloud computing domain from its internal private cloud

A ***community*** cloud shares computing resources across several organizations, and can be managed by either organizational IT resources or third-party providers.

- Cloud providers provide cloud services to the cloud users and cloud users uses *pay-as-you-go model* to use these services.
- The cloud users develop their product using these services and deliver the product to the end users.



ADVANTAGES OF CLOUD COMPUTING

1. Flexibility/Elasticity:
- Users can access computing resources as and when needed, without any human interaction.
- Cloud users can use the resources on demand basis and pay as much as they use

2. Scalability Of Infrastructure.
- New nodes can be added or removed from the network as can physical servers, with limited modifications to infrastructure set up and software. Cloud architecture can scale horizontally or vertically according to the users requirements.

3. Broad Network Access.
- promotes use of heterogeneous platforms (like mobile phones, laptops, and PDAs).

4. Location Independence.
- Cloud interfaces are location independent and they can be accessed by Web services and Web browsers, so that no knowledge about exact location of the user is required which also gives high level of abstraction to the users data.

5. Unlimited Storage.
- Storing information in the cloud gives almost unlimited storage capacity. Hence no more need to worry about running out of storage space or increasing current storage space availability.

6. Easy Access to Information.
- Once registered in the cloud environment one can access the information from any location provided, there is an Internet connection.

7. Economies of Scale and Cost Effectiveness.
- Cloud implementations tend to be as large as possible in order to take advantage of economies of scale.
- Large cloud deployments can often be located close to cheap deployment to lower cost.
- It does not require upfront investment and much capital expenditure as users may pay and use or pay for services and capacity as they need them.

8. Backup and Recovery.

- Most cloud service providers are competent enough to handle recovery of information which makes the entire process of backup and recovery much simpler than other traditional methods of data storage.

9. Reliability

- It improves the use of multiple sites which makes cloud computing suitable for business continuity and disaster recovery.

10. Sustainability

- It improves resource utilization and makes the cloud environment more efficient.

Questions

1. **(a)** Draw the diagram for the OSI Model. [7]
(b) Describe the following protocols as they are related to TCP/IP suite:
(i) TCP [2]
(ii) IP [2]
(iii) HTTP [2]
(c) State which layer each of the protocols in **(b)** belong to. [3]

2. **(a)** Explain the term *Domain Name System* (DNS). [2]
(b) Distinguish between *private IP* and *public IP*. [2]

3. Describe the layer in the protocol stack which is responsible for finding a communication partner on the network

4. Explain the activities performed at the Presentation layer

5. List and explain the Presentation layer protocols

6. Explain the function of the Session layer

7. List and explain Session layer protocols

8. Describe the layer which is responsible for multiplexing data from upper layers and placing the data into a segment

9. Windowing is performed at the Transport layer. Explain what windowing is.

10. Describe the primary function of the Network layer

11. What are the two parts to a network address?

12. The Data Link layer is split into two sublayers. Name the sublayers.

13. a) Which layer is responsible for creating and disconnecting virtual circuits
b)If your network diagnostic tool identifies a problem with the logical addressing, what layer of the OSI model would you be troubleshooting.
c) What layer would you troubleshoot when no link connectivity is detected
d) If two network cards were suspected of having the same MAC address, what layer would you troubleshoot to determine the conflict.

15. One of your nodes requests a window size of 1. This is having adverse effects on the network and you need to change it. What layer of the OSI model is responsible for this

16. List and explain cloud service models

17. Describe cloud types

Computer Architecture

1. Explain The Principle Of Operation Of Passive And Active Electronic Components

Electronic Components

- *Electronic components, both **active** and **passive**,* are lifeline of any printed circuit assembly.
- They play vital roles in the functioning of any electronic device.
- They are connected together, usually by soldering to a printed circuit board (PCB), to create an electronic circuit with a particular function.

Active Electronic Components

- *Active electronic components* are those that can control the flow of electricity.
- These include *transistors, vacuum tubes, silicon-controlled rectifiers (SCRs).*

Transistors

A transistor is a semiconductor device which is used in a number of functions including voltage regulation, amplification, switching, signal modulation, and oscillators.

Passive Electronic Components

- *Passive electronic components* are those that don't have the ability to control current by means of another electrical signal.
- These include capacitors, resistors, inductors, transformers, and diodes.

Resistor

- A Resistor is an electrical device that resists the flow of electrical current.
- It is a *passive device* used to control, or impede the flow of, electric current in an electric circuit by providing resistance, thereby developing a drop in voltage across the device.

Capacitor

- A capacitor is a passive electrical component that can store energy in the electric field between a pair of conductors called "plates".
- The process of storing energy in the capacitor is known as "charging".
- Capacitors are used in electronic circuits as energy storage devices.

Diode

- A diode is a one-way valve for electricity which allow flow of electricity in one direction.
- Most diodes have a painted line on one end showing the direction or flow with the negative side normally white.

Integrated Circuit (IC)

- Integrated Circuits are package of several complex circuits.

2. Describe The Von Neumann And Harvard Architecture

Von Neumann Architecture

Diagram A

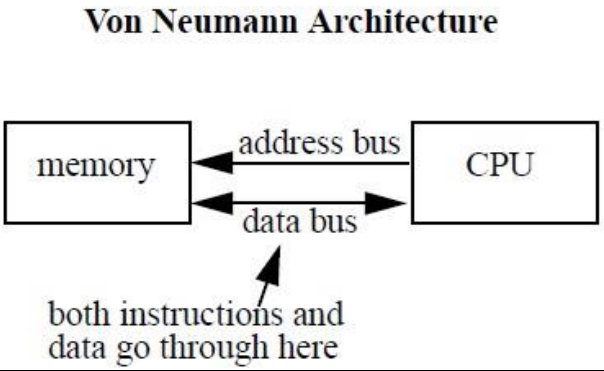
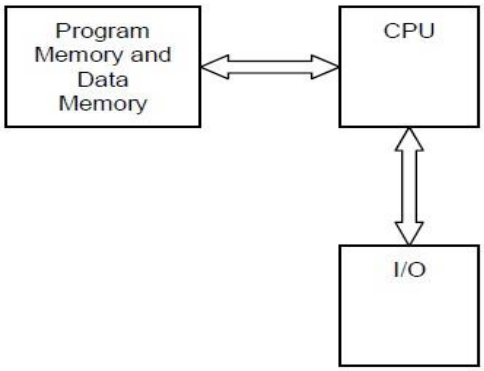
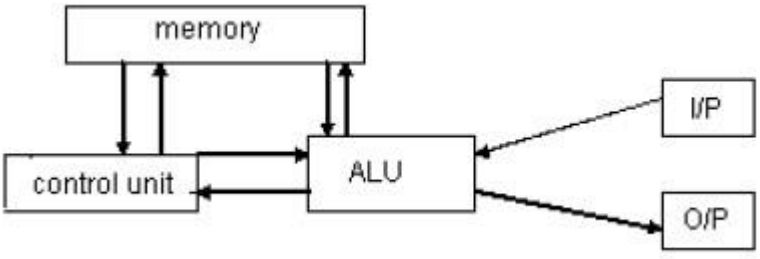


Diagram B

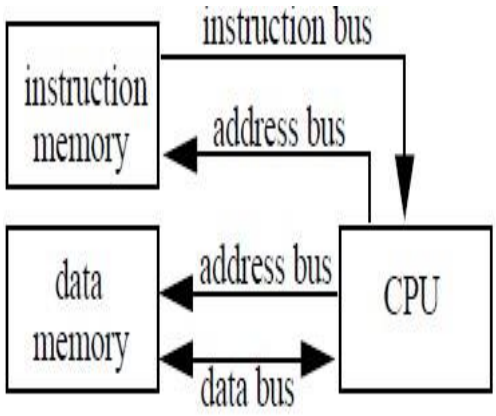
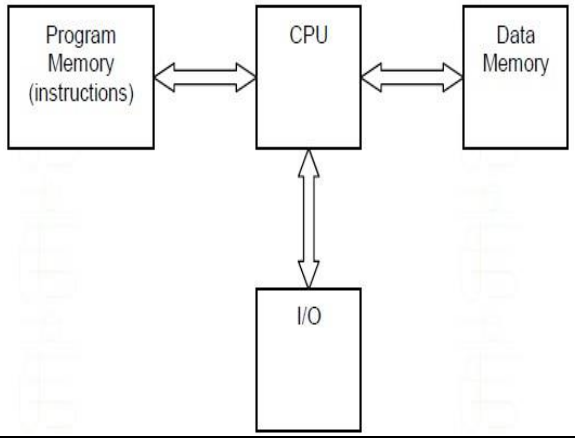


- The Von Neumann Computer also referred to as **stored-program** digital computer because keeps its programmed *instructions*, as well as its *data*, in read-write, random access memory (RAM), which makes the machines much more flexible.
- By treating those instructions in the same way as data, a stored-program machine can easily change the program, and can do so under program control.
- Has a single processor which follows a **linear sequence** of the fetch-decode-execute cycle. Therefore only one **job** processed **at a time with one set of data** and execution occurs in a sequential fashion from one instruction to the next, unless explicitly modified (jump instruction).
- Uses serial processing of instructions. Allows for **one instruction to be rea**d from memory or data to be read/written from/to memory at a time.
- Instructions and data are stored in the same memory and **share a communication pathway or bus** to the CPU. Because program memory and data memory cannot be accessed at the same time, throughput is much smaller than the rate at which the CPU can work. This constraint (problem) is called the Von Neumann **bottleneck** and directly impacts the performance of the system. This can however be solved by use of ***cache*** memory.
- The von Neumann architecture allows instructions and data to be mixed and stored in the same memory module and the contents of this memory are addressable by location only.
- Von Neumann architectures usually have a single unified cache
- The Von Neumann machine had five basic parts:- (i) **Memory** (ii) **ALU** (iii) **control unit** (iv) **Input equipment** & (v) **output equipment**



- Processor needs two clock cycles to complete an instruction i.e first to get the *instruction* from memory and decodes it second cycle to get the required *data* from memory.
- Pipelining the instructions is not possible with this architecture.

Harvard Architecture.

Diagram A	**Diagram B**
	

The Harvard architecture has the following characteristics:

- Stores *instructions* and *data* in separate memory, thus has physically separate storage for data and instructions
- Has separate data and instruction buses, allowing transfers to be performed simultaneously on both buses.
- Data and instructions are treated separately
- Uses pipelining whereby *Operand Fetch* and *Instruction Fetch* can be overlapped.
- Harvard Architecture would have separate caches for each bus.

3. Explain the use of buses

Memory Unit

Is the computer memory that temporarily stores the operating system, application programs and data currently use.

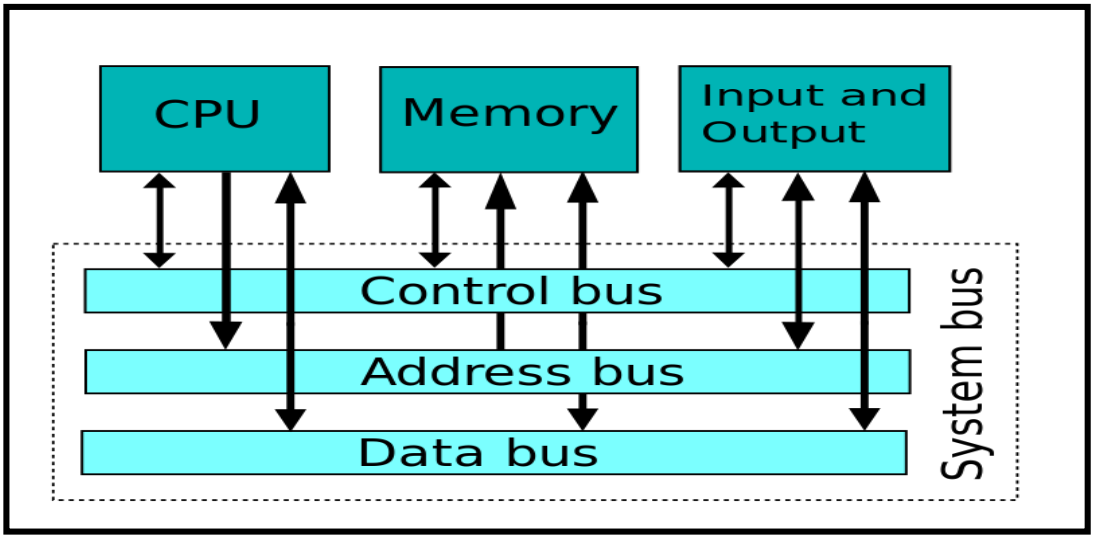
It used to store the following:

- Program instructions in current use;
- Data in current use;
- Parts of Operating System that are currently in use.

Some architectures have a Memory Unit (Main memory) which has two types: RAM and ROM.

Buses

- A bus is a pathway through which data and signals are transferred from one device to another.
- They are a set of parallel wires connecting two or more components of the computer.
- Buses can be internal or external, internal bus is generally referred to as **system** bus and this connect the **CPU, memory** and **I/O devices.**
- Each bus is a shared transmission medium, so that only one device can transmit along a bus at any one time.
- Multiple devices can be connected to the same bus



- *Data and control* signals travel in both directions between the processor, memory and I/O controllers.
- *Address* travel only one way along the address bus, the processor sends the address of an instruction, or of data to be stored or retrieved, to memory to an I/O controller.

Data bus:

- Used for carrying data from memory to the processor and between I/O ports.
- Comprises of either 8, 16, 32 or 64 separate parallel lines
- Provide a *bi-directional* path for data and instructions between computer components i.e the CPU can read data from memory and input ports and also send data to memory and output ports.
- The width of the bus determines the overall system performance e.g, if the data bus is 8 bits wide, and each instruction is 16 bits long, then the processor must access the main memory twice during each instruction cycle

Address bus:

- This a uni-directional bus (one way). The address is send from CPU to memory and I/O ports only.
- Used for transferring memory addresses from the processor when it is accessing main memory
- They are used to access memory during the read or write process
- The width of the address bus determines the maximum possible memory capacity of the computer.

Control bus:

- This is a bi-directional bus used for carrying control signals (Signals can be transferred in both directions).
- It transmits *command*, *timing* and specific *status* information between system components.
- They carry signals to enable outputs of addressed port and memory devices
- Control signals regulate activities on the bus.
- Typical control signals are:
  - ✓ Memory Read
  - ✓ Memory Write
  - ✓ I/O Read
  - ✓ I/O Write

- ✓ Interrupt Request
- ✓ Interrupt Grant
- ✓ Reset
- ✓ Ready hold

- **Timing signals**: indicate validity of data and information.
- **Command signals**: Specify operations to be performed
  - **Status signals**: Indicate state of data transfer request or status of a request or the status of request by a components to gain control of the system bus

4. Demonstrate The Use Of Logic Gates

- A logic gate is a device that produce signals of 1 or 0 when the input logic requirements are met and are used in manipulating binary information.
- A logic gate is a device (or electrical circuit) that performs one or more logical operations on one or more input signals.
- Logic gates are the building blocks of digital technology. - They can be used in applications like:
  - Building computer chips
  - Programming traffic signals
  - Chips for automatic alarm systems
  - Chips for automated control systems

Main Logic Gates

The main logic gates are:

(a) OR gate
(b) AND gate
(c) NOT gate
(d) NOR gate
(e) NAND gate
(f) Exclusive OR gate (XOR)
(g) Exclusive NOR gate (XNOR)

Logic gates are used with **truth tables**.

- A ***truth table*** is a table that describes the behaviour of a logic gate, which shows how a *logic* circuit's output responds to various combinations of the inputs, using ***logic* 1** for true and ***logic* 0** for false.
- It lists the value of the output for every possible combination of the inputs

The number of rows in a truth table shows the number of *possible combinations* of the inputs of a particular circuit. The number of rows for each gate is found using the following formulae:

Combinations = 2^n , where *n* being the number of inputs in the gate or circuit.

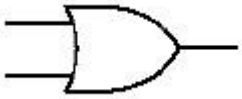
For example, a gate or circuit has the following possible combinations corresponding to the number of input:

- 1 input = 2^1 = 2 combinations
- 2 inputs = 2^2 = 4 combinations
- 3 inputs = 2^3 = 8 combinations
- …..

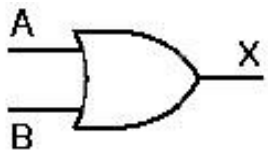
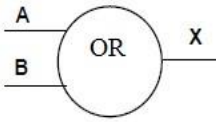
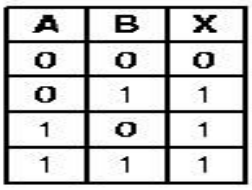
Graphical Representation of Gates and their Truth Tables

Each logic gate has its own unique diagram. Even if the name of the gate is not written, one knows what it stands for because of the shape. The following are the logic gates and their shapes in standard form.

(a) OR gate



This represents two inputs entering the gate and one output from the gate. The inputs can be represented by **any** alphabetic characters, e.g. A and B, while the output can be X, given as follows:

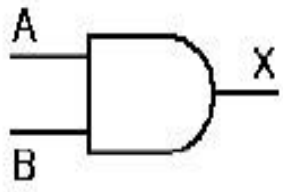
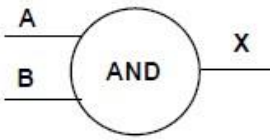
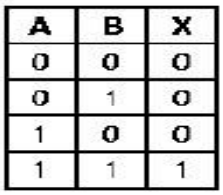
Logic Gate Diagram		**Truth table**
Standard Form	**General Form**	
		

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

- X = A **OR** B can also be represented as X = A+B
- The output (X) is **true** if the **INPUT A OR INPUT B** are **true.**
- It is true if **either one** of the two is true or **both**

(b) AND gate

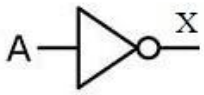
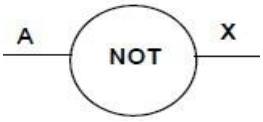
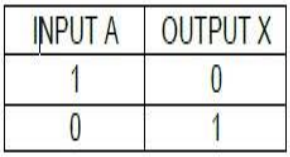
This is represented as follows:

Logic Gate Diagram		**Truth table**
Standard Form	**General Form**	
		

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

- X = A **AND** B. or X= A.B
- The output (X) is only **true** if the **INPUT A AND INPUT B** are both **true**.
- X = 1 if and only if A and B are all equal to 1

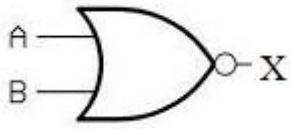
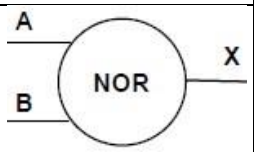
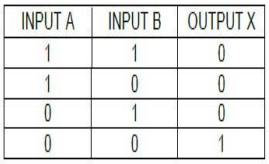
(c) NOT gate

Logic Gate Diagram		**Truth table**
Standard Form	**General Form**	
		

INPUT A	OUTPUT X
1	0
0	1

- A' or $\overline{A}$
- The **NOT** gate has **only** one input and one output. The input is negated. Thus if input is 1, output is 0, and vice versa.
- The output (X) is **true** when the **INPUT A** is **NOT TRUE**. The output (X) is **False** when the **INPUT A** is **TRUE.**

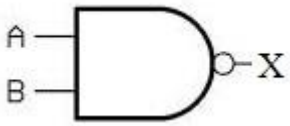
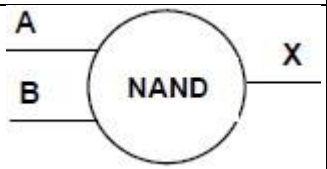
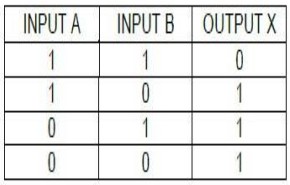
(d) NOR gate

Logic Gate Diagram		**Truth table**
Standard Form	**General Form**	
		

INPUT A	INPUT B	OUTPUT X
1	1	0
1	0	0
0	1	0
0	0	1

- This is an **OR** gate with the output X inverted.
- The output (X) is **true** if **NOT** (**INPUT A OR INPUT B**) are **true.**
- Thus X = NOT (A or B)

(e) NAND gate

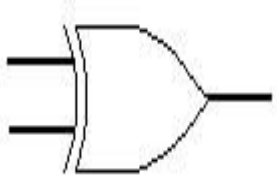
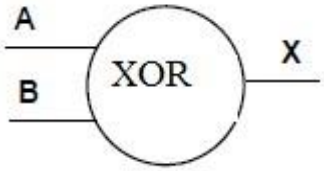
Logic Gate Diagram		**Truth table**
Standard Form	**General Form**	
		

INPUT A	INPUT B	OUTPUT X
1	1	0
1	0	1
0	1	1
0	0	1

- This is an **AND** gate with the output X inverted.
- The output is **true i**f **INPUT A AND INPUT B** are **NOT** both **True.**
- It translates to NOT (A and B)

(f) Exclusive OR gate (XOR)

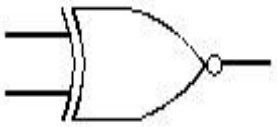
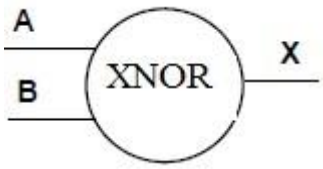
In this gate, the output is 1 (T) if either, but not both, of the inputs are 1 (T). The output is 0 (False) if both inputs are 0(False) or if both inputs are 1(True). In other words, the output is 1 if the inputs are different, but 0 if the inputs are the same.

Logic Gate Diagram		**Truth table**
Standard Form	**General Form**	
		

INPUT A	INPUT B	OUTPUT X
1	1	0
1	0	1
0	1	1
0	0	0

(g) Exclusive NOR Gate (XNOR)

The *XNOR (exclusive-NOR) gate* is a combination XOR gate followed by an inverter. Its output is 1 if the inputs are the same, and 0 if the inputs are different.

Logic Gate Diagram		**Truth table**
Standard Form	**General Form**	
		

INPUT A	INPUT B	OUTPUT X
1	1	1
1	0	0
0	1	0
0	0	1

5. Describe The Functions Of Processor Components

1. Arithmetic and Logic Unit

Responsible for carrying out operations on data, like calculations. It consists of the **Arithmetic Unit** and the **Logic Unit**

Arithmetic Unit

Responsible for basic arithmetic functions such as: Addition, Subtraction, Multiplication, Division, etc

Logical Unit

It perform logical operations like comparing two data items to find which data item is > ,= ,< the other, etc

The ALU works together with the *accumulator* register, which temporarily stores data being processed and the results of processing.

The ALU performs the following:

- Carries out all arithmetic.
- Carries out logic operations.
- Acts as gateway to and from the processor

2. Control unit

- It manages the execution of instructions by running the clock.
- It coordinates and controls all operations of computer system.
- It performs the following:
    - Fetches the next instruction to be executed
    - Decodes instructions
    - Manages execution of instructions
    - Executes decoded instructions
    - It carries out the **Fetch-Execute Cycle.**

3. Registers:

- This is a **high-speed storage** area **in the CPU** used to **temporarily** hold small units of program instructions and data immediately before, during and after execution by the CPU.
- It is a small amount of storage available on the CPU whose contents can be accessed more quickly than storage available elsewhere
- Provides the fastest way for a CPU to access data.
- Most modern computer architectures operate by moving data from main memory into registers, operate on them, then move the result back into main memory
- Register size determines how much information it can store: i.e., can be one, two, four or eight byte register
- The processor contains a number of special purpose registers (which have dedicated uses) and general purpose registers (which may be used for arithmetic function and are a sort of "working area")
- The **main types** registers (special purpose registers) found in the Von Neumann Machine are as given below:
    - ✓ program counter
    - ✓ memory address register
    - ✓ memory data register/memory buffer register
    - ✓ current instruction register
    - ✓ index register

Program Counter (PC)

- Contains the address of the next instruction to be fetched/executed
- PC holds address of next instruction
- this register is automatically incremented so that it always holds the memory address of the next instruction
- Keeps track of whereabouts of the next program in the memory.
- After one instruction has been carried out, the PC will be able to tell the processor the whereabouts of the next instruction.
- The PC is also called the Sequence Control Register (SCR) as it controls the sequence in which instructions are executed.
- During program execution, the PC:
    - stores the address of the next instruction to be executed
    - Its content is automatically incremented after the address is read, or
    - Its content is altered to specific address if instruction is a **jump** instruction

Memory address register (MAR)

- Hold the memory address that contains either the next piece of data or an instruction that is to be used.
- It holds the address of a memory location from which data will be read from or written to.
- Specifies the address for the next read or write
- This is where the address that was read from the PC is sent.

Memory Data Register (MDR) also called Memory buffer register (MBR)

- Used to store data which has been read from or is ready to write to memory.
- It acts like a buffer and holds anything that is copied from the memory ready for the processor to use it
- It contains data written into memory or receives data read from memory

Current instruction register (CIR)

- it holds the instruction that is to be executed
- Contains both the operator and operand of the current instruction - For example the instruction ***MOVE 100,#13***
- "MOVE" is the operator, and "100" and "#13" are the operands.
- It stores an instruction while it is being decoded/executed/carried out
- Its contents change when an instruction from memory has been placed in MDR, and then it is copied from MDR to CIR.

The Status Register (SR).

- This contains status bits which reflect on the results of an instruction. For example, if there is an error in the operation (such as an overflow) this will be recorded in the status register. The Status Registers also store data about interrupts

Index register

- It is a register used for modifying operand addresses during program execution,
- Used in performing vector/array operations.
- Used for indirect addressing where an immediate constant (i.e. which is part of the instruction itself) is added to the contents of the index register to form the address to the actual operand or data

General Purpose Registers Accumulator

- A general purpose register used to accumulate results of processing
- Is where the results from other operations are stored temporarily before being used by other processes.
- Used for performing arithmetic functions

6. Identify Factors Affecting Processor Speeds

Processor Performance

The traditional processor's performance is affected by these four main components:

(a) Clock Speed

- The processor contains a timing device known as the clock which determines the timing of all operations.
- This sends out signals at a given interval, and all processes within the computer will start with one of these pulses.

- A process may take any amount of time to complete, but it will only start on a pulse.
- A processor with a faster clock speed will perform faster, since more pulses will be sent out in the same time frame.
- A faster clock increases the speed of the processor and/or memory but not the peripherals
- The clock speed is measured in Hertz, with modern processors typically Gigahertz.

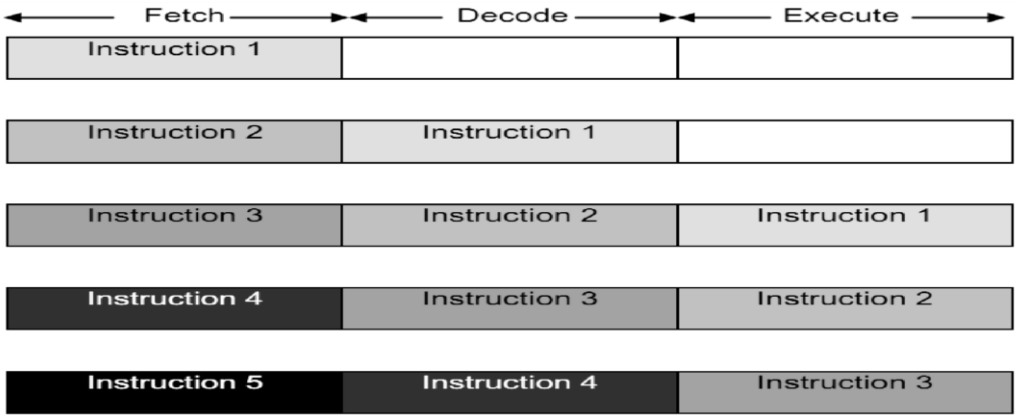
(b) Word Size
- The word size of a computer is the number of bits it can process at a time.
- Increasing the word length of the registers benefits programs with many numerical calculations - Word length is determined by size of registers
- Bits are grouped into words, the length of a word varies but it is typically, 8, 16, 32, 64 or 128 bits.
- Obviously if the processor is able to deal with more bits at a time then it is going to perform better.

(c) Bus Width
- The addresses of data, and the data itself, is transmitted along buses inside the computer.
- The width of the bus is the number of bits it can store / carry.
- A wider data bus will allow more data to be sent at a time, and therefore the processor will perform more efficiently.
- A wider address bus will increase the number of memory addresses a computer may use.
- For example an 8 bit bus will only allow a value between 0 and 255 to be transmitted at a time.

(d) Architecture
- The architecture of a processor will affect its performance,
- A better designed processor will perform better than a different processor.

7. Explain The Importance Of Pipelining

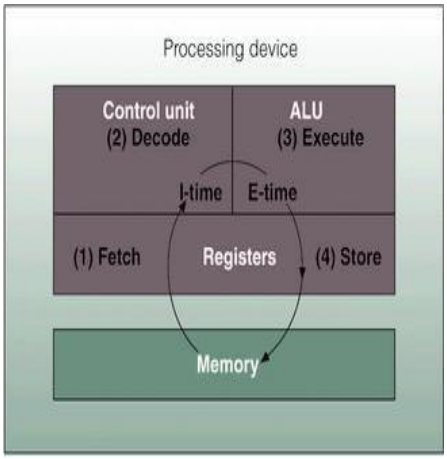
Pipeline Processing
- It is a technique which allows the overlapping of the fetch-decode-execute cycle for different instructions.
- A parallel processing architecture in which several processors are used, each one doing a different part of the fetch, decode, execute cycle, so the fetch-decode-execute cycle is staggered.
- The processor is split up into three parts (fetch, decode, execute), each of which handles one of the three stages.
- Each part is called a line, where each single line is a pipeline.



As long as the pipelines can be kept full, it is making best use of the CPU.
In pipelining, three instructions are dealt with at the same time. This reduces the execution time considerably.
Once jump instructions are introduced the problem arises that the wrong instructions are in the pipeline waiting to be executed, so every time the sequence of instructions changes, the pipeline has to be cleared and the process started again.
A non-pipeline architecture is inefficient because some CPU components (modules) are idle while another module is active during the instruction cycle. Pipelining does not completely cancel out idle time in a CPU but making those modules work in parallel improves program execution significantly.
Processors with pipelining are organized inside into stages which can semi-independently work on separate jobs. Each stage is organized and linked into a 'chain' so each stage's output is fed to another stage until the job is done. This organization of the processor allows overall processing time to be significantly reduced

8. describe the fetch-decode execute cycle

The Fetch-Decode-Execute Cycle



Step 1. Fetch instruction: In the instruction phase, the computer's control unit fetches the next instruction to be executed from main memory. Microprocessor gets software instruction telling it what to do with data.
Step 2. Decode instruction: Then the instruction is decoded so that the central processor can understand what is to be done. Microprocessor determines what the instructions mean. At this stage, the computer produces signals which control other computer components like the ALU.
Step 3. Execute the instruction: In the execution *The Fetch-Execute Cycle* phase, the ALU does what it is instructed to do, making either an arithmetic computation or a logical comparison. Microprocessor performs the instruction (cause instruction to be executed).
Step 4. Store results: Then the results are stored in the registers or in memory.
Step 3 & 4 are called the execution phase. The time it takes to complete the execution phase is called the EXECUTION TIME (E-time).
After both phases have been completed for one instruction, they are again performed for the second instruction, and so on.

The Fetch-Decode Execute Cycle

There are three stages of the machine cycle in a Von Neumann architecture, which are: fetch, decode and execute stages.

Fetch

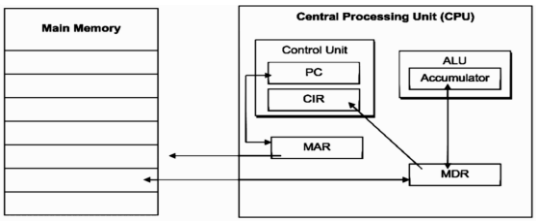
- The PC stores the address of the next instruction to be carried out (*because instructions are held sequentially in the memory, the value in the PC is incremented so that it always points to the next instruction.*)
- When the next instruction is needed, its address is copied from the PC and placed in the MAR
- The data which is stored at the *address* in the MAR is then copied to the MDR
- Once it is ready to be executed, the executable part if the instruction is copied into the CIR

Decode

- The instruction in the CIR can now be split into two parts, the *address* and the *operation*
- The address part can be placed in the MAR and the data fetched and put in the MDR.

Execute

- The contents of both the memory address register and the memory data register are sent together to the central processor.
- The ALU will keep referring back to where the data and instructions are stored, while it is executing them, the MDR acts like a buffer, storing the data until it is needed
- The CU will then follow the instructions, which will tell it where to fetch the data from, it will read the data and send the necessary signals to other parts of the computer.



In short fetch-decode-execute cycle operates in the following way:

- Load the address that is in the program counter (PC) into the memory address register (MAR).
- Increment the PC by 1.
- Load the instruction that is in the memory address given by the MAR into the MDR
- Load the instruction that is now in the MDR into the current instruction register (CIR).
- Decode the instruction that is in the CIR.
- If the instruction is a jump instruction then
  - Load the address part of the instruction into the PC
  - Reset by going to step 1.
- Execute the instruction.
- Reset by going to step 1.

9. Identify Types Of Interrupts

Interrupt:

- It is a signal generated by a device when it requires processor attention.
- An interrupt is a signal send to the processor by a peripheral or software for attention to be turned to that peripheral/software, thereby causing a break in the execution of a program, e.g. printer out of paper.

Types of interrupts

- Input / output interrupt: generated by the I/O devices when transfer is complete or when there is an error in transmission, e.g. disk full, printer out of paper, etc.
- Interrupts generated by running process: process may need more storage or to communicate with the operator
- Timer interrupts: generated by the processor clock, e.g. control being transferred to another user in a *time sharing* system, or the processor want to perform a routine task which is done at that exact time daily.
- Program check interrupts: caused by errors in a program, e.g division by zero
- **Machine check interrupts:** Caused by malfunctioning hardware**.**

Sources of interrupts

- power failure/system failure
- peripheral e.g. printer (buffer empty)/hardware
- clock interrupt
- user interrupt e.g. new user log on request
- software

Interrupt Service Routine (Handler):

- is a small subprogram that handles the interrupts.

Interrupt Handling

Interrupts have different **priorities** to help the processor decide in case two interrupts are received simultaneously, which one is more important to execute first.
There are four levels of priority, which are (highest priority order):

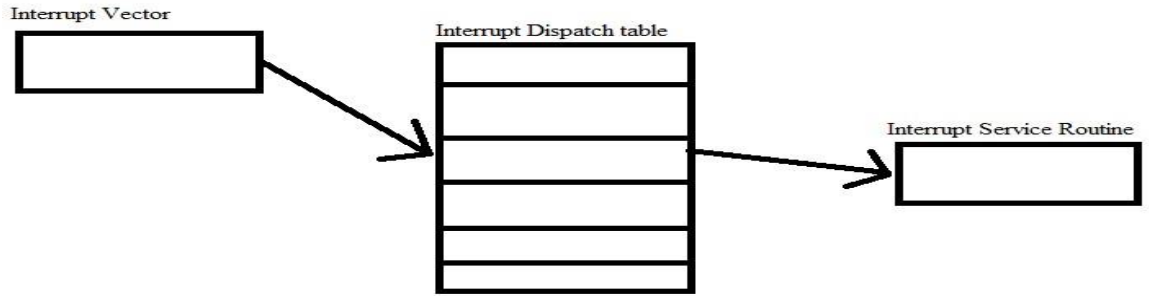
1. *Hardware Failure*: can be caused by power failure or memory parity error.
2. *Program Interrupts*: Arithmetic overflow, division by zero, etc
3. *Timer Interrupts*: generated by the internal clock
4. *I/O Interrupts:* generated by the I/O devices

At the end of each Fetch-Execute cycle, the contents of the interrupt registers are checked for interrupt; if there is an interrupt:

a) The current fetch-decode-execute cycle is completed
b) The contents of the PC and other registers will be stored away safely in a stack.
c) Source of interrupt is identified, Interrupts with a lower priority are disabled.
d) The start address of the interrupt handler is loaded into the PC.
e) The interrupt handler is executed.
f) Interrupts are enabled again, and the cycle will restart with any further interrupts.
g) Contents of PC and other registers are “popped” from the stack and restored.
h) The program resumes with the next step in its cycle.

Vectored Interrupts

- An interrupt mechanism used to identify which interrupt handler to call for which interrupt.
- Assign specific number to each interrupt called an interrupt vector.
- Address of interrupt service routines are stored in an array (known as interrupt dispatch table) and the interrupt vector is used as a subscript to this array.



10. Justify Why Computers Use Interrupts

Why interrupts are used in a computer system

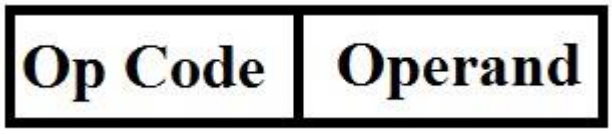
- to obtain processor time for a higher priority task
- to avoid delays
- to avoid loss of data
- as an indicator to the processor that a device needs to be serviced
- it allows the processor to go in a very low-power mode when no computations are to be run, and to be 'awaken' only when any external data is ready
- allows computer to shut down if the power off interrupt predicts loss of power, saving data in time
- to perform a task which is of higher importance when the processor is busy with an another task.

11. Explain Addressing Modes

ADDRESSING MODES

- The ways in which a computer calculate addresses holding the source and/or destination of the data being processed in a particular instruction.

Each assembly language instruction has the following structure:



Op-code (operator):

- Is the part that represent the operations that the computer can understand and carry out. It is the mnemonic part of the instruction/that indicates what it is to do/code for the operation. E.g. ADD.

Operand:

- it is the address field in an instruction that holds data to be used by the operation given in the opcode, e.g. in ADD 12, "12" is the operand
- is the data to be manipulated, It can be the address of the data, or just the data.

Addressing

- refers to the way in which data can is represented.

1. **Symbolic addressing:** the use of characters to represent the address of a store location
2. **Effective Address**: the actual address of operand to be used by the instruction.

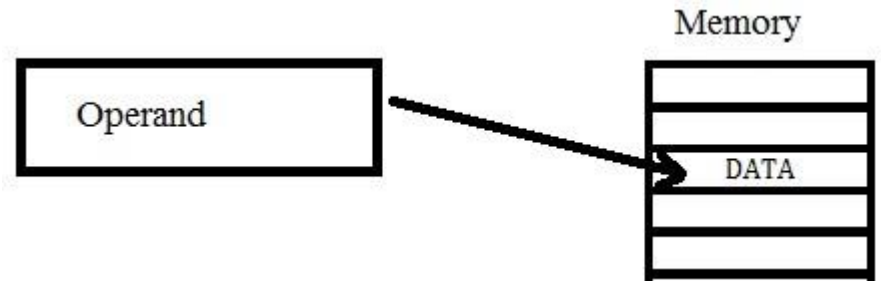
The most common addressing modes are: *direct, indirect, indexed, relative and immediate addressing.*

Immediate Addressing

- This is when the value in the instruction is not an address at all but the actual data (constant to be used in the program).
- The data to be operated on is held as part of the instruction format.
- The data to be used is stored immediately after the op code for the instruction i.e the operand field actually contains the data
- e.g: LDA #&80: Means that Load the hexadecimal value of 80 into the accumulator register.
- This is very simple, although not often used because the program parameters cannot be changed.
- This means that the data being operated on can't be adjusted and only uses constants.
- Can be used to initialize constants.

2. Direct Addressing

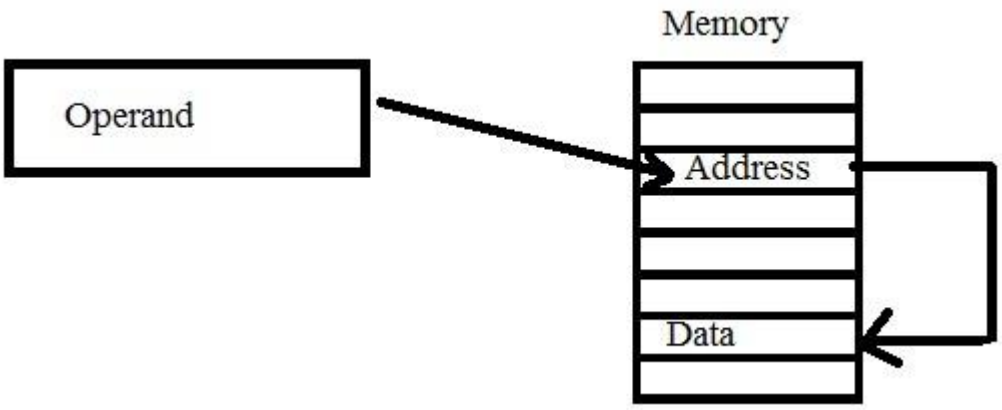
- The address in the instruction is the address to be used to get to the operand.
- The operand gives the address of the data to be used in the program.



- It requires **one** memory reference to read the operand from the given location
- The address given in the instruction is the one that contains the data to be used in the operation **without any modification.**
- e.g In the instruction **ADD 23**, first go to memory address 23 which stores the instruction to be executed.
- It provides only a limited address space
- It is very simple, although does not make best use of memory
- It is slow as too much memory is used

3. Indirect Addressing

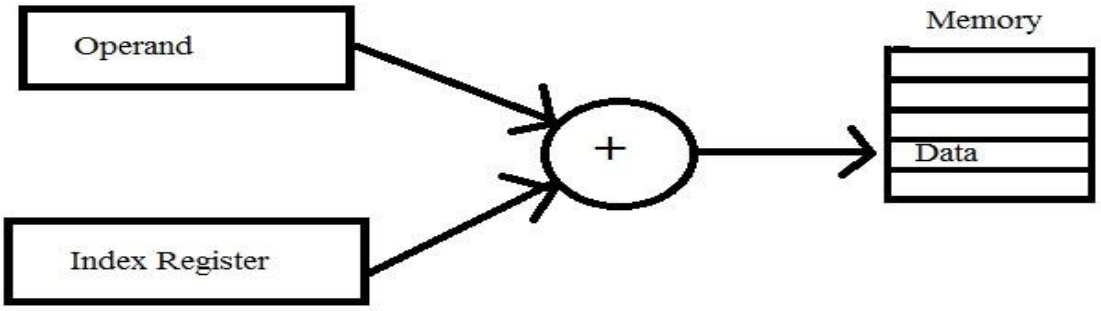
- In this mode of addressing, the address given in the instruction holds the address of where the data is stored.
- This is whereby the real address is stored in the memory so the value in the address part of the instruction is pointing to the address of the data.
- The address of data in memory is held in another memory location and the operand of the instruction holds the address of this memory location.



- It is MOSTLY used when accessing areas of memory that are not accessible using the space available for the address in the instruction code
- E.g ADD 23, go to memory address 23 and get another memory address, e.g 32, where the actual instruction will be stored
- This method is useful because the amount of space in a location is much bigger than the space in the address part of the instruction.
- It gives flexibility as the original program does not need to be altered if the position of the routines (sub-programs) change.
- Therefore we can store larger addresses and use more memory.
- It is used where memory is larger than can be accessed by address in instruction
- It is also used when one wants to allow full size of register to be used for address
- used if memory locations which are 32 bits are used and thus allowing more memory to be accessed

Indexed Addressing

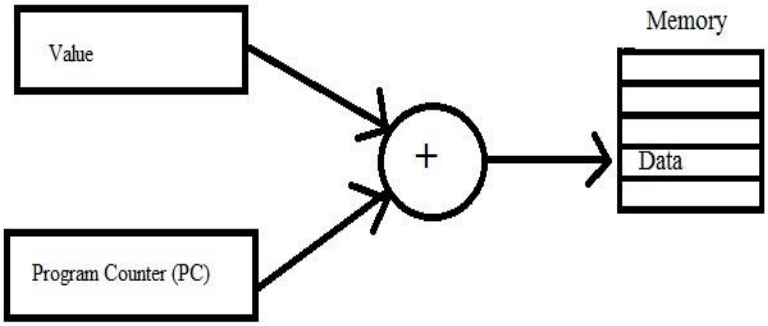
- The address part of the instruction is added to a value held in the index register.
- It is where the actual address is found by adding a displacement to the base address.



- The instructions specify two registers. The processor then adds contents of these two registers to get the *effective* address.
- One of the registers is an address register and it holds the base address while the other one is a data register or the displacement or index register - Jmp +10: *branch to instruction 10 bytes on.*
- Used when a number of contiguous locations need to be accessed in order e.g. contents of array
- Used **when** address in instruction does not change (need not to be altered-like constants), only contents of IR need to be changed

Relative Addressing

- The same as Indexed Addressing except that the PC replaces the Index Register.
- E.g Load R_i, X (PC)
- This loads register R_i with the contents of the memory location whose address is the sum of the contents of the PC and the value X.



- It begins from a fixed point, and all addresses are relative to that point.
- allows a real address to be calculated from a base address by adding the relative address
- relative address is an offset and can be used for arrays and/or branching instructions
- it adds the PC contents to the base address to get the effective address
- It is appropriate when the code is going to be loaded at a random place in memory to be executed.
- Use to refer to jump instructions

12. Construct Software Architectural Design For A Given Scenario

Software Architecture

- The architecture of a system describes its major components, their relationships (structures), and how they interact with each other.
- The structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

The Importance of Architecture

– provides a **communication** among stakeholders
– captures **early design** decisions
– acts as a **transferable abstraction** of a system
– defines **constraints** on implementation
– dictates **organizational** structure
– is analysable and a vehicle for **predicting** system qualities
– makes it easier to reason about and **manage change**
– helps in evolutionary **prototyping**
– enables more accurate **cost** and **schedule** estimates

Architectural Drivers

- These the design **forces** that will influence the early design decisions the architects have to make
- Architectural drivers are not all of the requirements for a system, but they are an early attempt to identify and capture those requirements, that are **most influential** to the architect making early design decisions.

Functional Requirements
specify what the software needs to do, they relate to the **actions** that the product must carry out in order to satisfy the fundamental reasons for its existence.

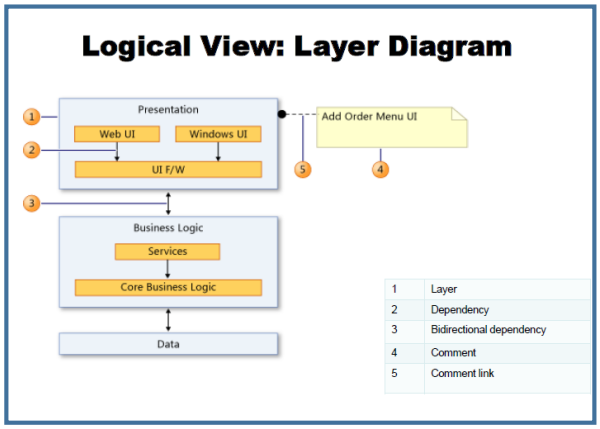
1. **Business Level**: defines the objective/goal of the project and the measurable business benefits for doing it.
2. **User Level**: user requirements are written from the user's point-of-view.
3. **System Level**: defines what the system must do to process input and provide the desired output.

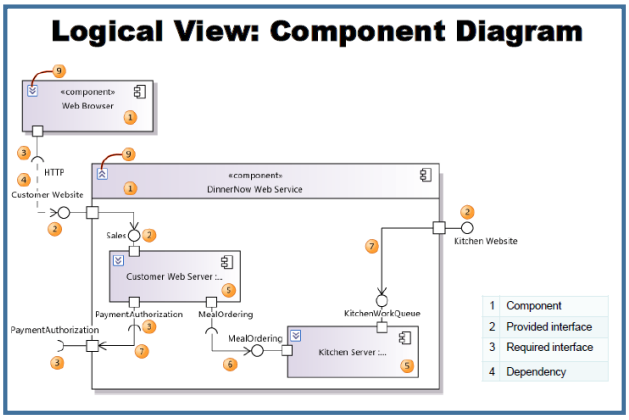
MoSCoW Method:
– **M - MUST**: Describes a requirement that *must* be satisfied in the final solution for the solution to be considered a success.
– **S - SHOULD**: Represents a high-priority item that *should* be included in the solution if it is possible. This is often a critical requirement but one which can be satisfied in other ways if strictly necessary.
– **C - COULD**: Describes a requirement which is considered desirable but not necessary. This will be included if time and resources permit.
– **W - WON'T**: Represents a requirement that stakeholders have agreed will not be implemented in a given release, but may be considered for the future.

Architects use a number of tools to model the architecture of a system at different view levels. These include

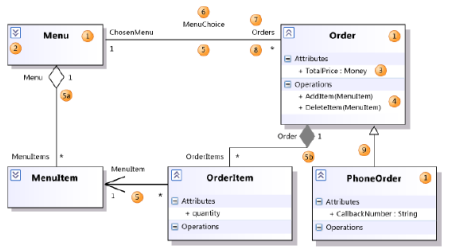
- Layer diagrams
- Component Diagrams
- Sequence Diagrams
- Class Diagrams
- Use Case diagrams
- Activity Diagrams
- Deployment Diagram

The following diagrams are for an **Online Catering Service** System



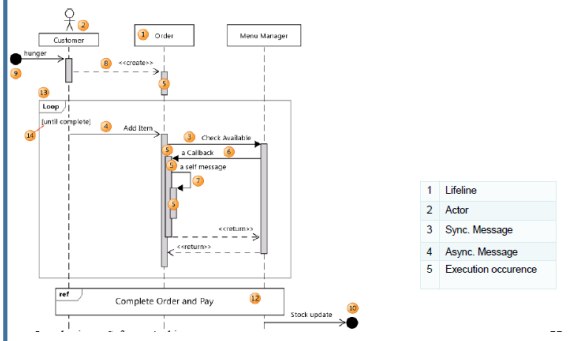


Logical View: Class Diagram



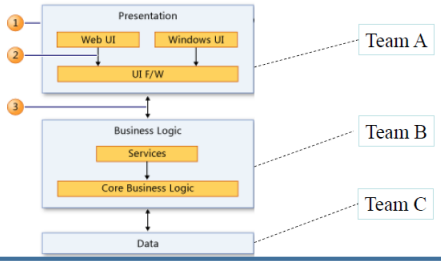
1	Class
3	Attribute
4	Operation
5	Association
5a	Aggregation
5b	Composition
9	Generalization

Logical View: Sequence Diagram

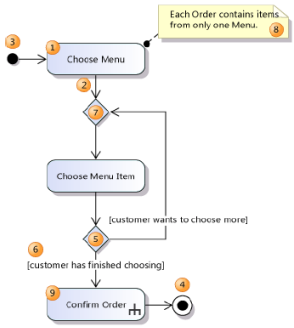


1	Lifeline
2	Actor
3	Sync. Message
4	Async. Message
5	Execution occurence

Development View: Layer Diagram

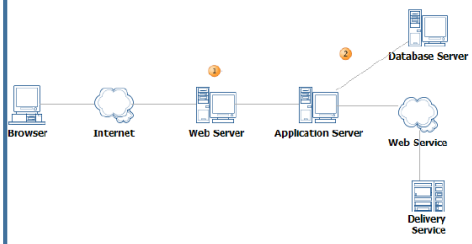


Process View: Activity Diagram



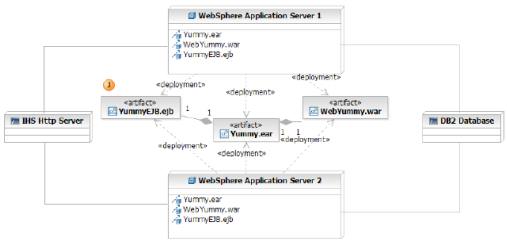
1	Action
2	Control flow
3	Initial node
4	Final node
5	Decision node
7	Merge node

Physical View: Deployment Diagram



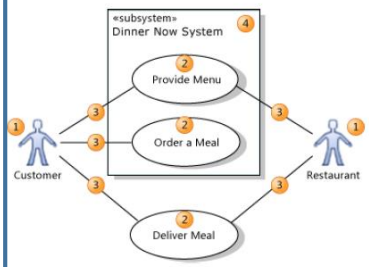
1	Node
2	Association

Physical View: Deployment Diagram



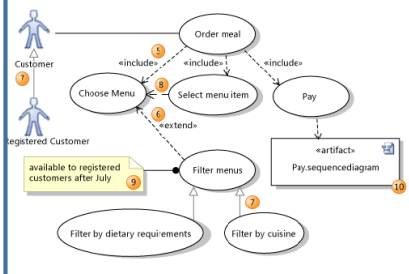
3	Artifact

Scenarios View: Use Case Diagram



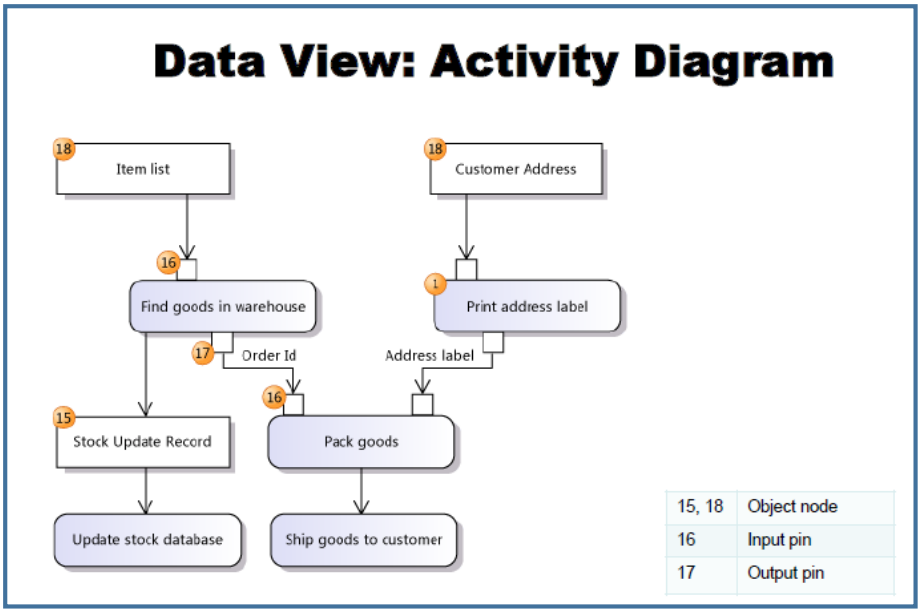
1	Actor
2	Use Case
3	Association
4	Subsystem

Scenarios View: Use Case Diagram



5	Include
6	Extend
7	Inheritance
8	Dependency

maregat@mbizohighschool.ac.zw



Questions

(a) (i) Draw a diagram to represent the Von Newmann Architecture. [4]
(ii) Explain the role of an accumulator in the Fetch Execute Cycle. [2]
(iii) With the aid of a diagram, illustrate the Fetch Execute Cycle. [4]
(b) Compare features of the Von Newmann Architecture and the Havard Architecture. [4]
6 (a) The following is a logic circuit
Write a logic statement that describes the above logic circuit. [3]
(b) Use a diagram to represent the XOR gate. [3]

Security and Ethics

1. Explain The Difference Between Data Privacy And Integrity

- **Information/data privacy** refers to the right of individuals and companies to deny or restrict the collection and use of data or information about them (information is not revealed without permission.)
- Security means that data is confidential and is not open for unauthorized parties.

The three most important goals for the *security* are:

- **integrity**, which means that information cannot be changed during transmission
- **authentication**, which occurs when an identity is established
- **confidentiality**, which means that the information is secure and stays confidential during transmission

2. Analyse Common Threats And Vulnerabilities On Computer Systems

A **computer threat or risk** is any event or action that could cause a loss of or damage to computer hardware, software, data, information, or processing capability

- **computer crime**- Any illegal act involving a computer
- **cybercrime** refers to online or Internet-based illegal acts.
- **hacker**, a computer enthusiast, who accesses a computer or network illegally.
- A **cracker** is someone who accesses a computer or network illegally but has the intention of destroying data, stealing information, or other malicious action. *Both hackers and crackers have advanced computer and network skills.*
- A **script kiddie** has the same intent as a cracker but does not have the technical skills and knowledge, uses prewritten hacking and cracking programs to break into computers.
- **corporate spies** have excellent computer and networking skills and are hired to break into a specific computer and steal its proprietary data and information, or to help identify security risks in their own organization. (used for *corporate espionage* whereby unscrupulous companies hire corporate spies, to gain a competitive advantage.)
- **Unethical employees** may break into their employers' computers for a variety of reasons eg to exploit a security weakness,seek financial gains from selling confidential information or *disgruntled* employees may want revenge.
- A **cyberextortionist** is someone who uses e-mail as a vehicle for extortion by sending an organization blackmailing e-mail message, claim a sum of money in exchange.
- **cyberterrorist** is someone who uses the Internet or network to destroy or damage computers for political reasons. The cyberterrorist might target the nation's air traffic control system, electricity-generating companies, or a telecommunications infrastructure. The term, ***cyberwarfare,*** describes an attack whose goal ranges from disabling a government's computer network to crippling a country.

3. **Identify Sources Of Vulnerability**
4. **Evaluate Tools Used To Eliminate Vulnerabilities**
5. **Identify Risks To Computer Systems**

1. Internet and Network Attacks

Information transmitted over networks has a higher degree of security risk than information kept on an organization's premises.

An **online security service** is a Web site that evaluates your computer to check for Internet and e-mail vulnerabilities, it then provides recommendations of how to address the vulnerabilities.

a) **Malware** (*malicious software*) - program that act without a user's knowledge and deliberately alter the computer's operations eg viruses, worms, Trojan horses, and rootkits are classified as

1. **Virus** is a potentially damaging computer program that affects, or infects, a computer negatively by altering the way the computer works without the user's knowledge or permission.
2. **Worm** is a program that copies itself repeatedly, for example in memory or on a network, using up resources and possibly shutting down the computer or network.
3. **Trojan horse** is a program that hides within or looks like a legitimate program. Unlike a virus or worm, a Trojan horse does not replicate itself to other computers and certain condition or action usually triggers the Trojan horse.
4. **Rootkit** is a program that hides in a computer and allows someone from a remote location to take full control of the computer. Once the rootkit is installed, the rootkit author can execute programs, change settings, monitor activity, and access files on the remote computer.

Safeguards against Malwares

- Methods that guarantee a computer or network is safe from malwares simply do not exist, therefore users have to take several *precautions*, to protect their home and work computers and mobile devices from these malicious infections.
  1. Do not start a computer with removable media inserted in the drives or plugged
  2. Never open an e-mail attachment unless you are expecting the attachment *and* it is from a **trusted source. (**A **trusted source** is an organization or person you believe will not send a virus infected file knowingly)
  3. Users should install an **antivirus program** and update it frequently, an **antivirus program** protects a computer against viruses by identifying and removing any computer viruses found in memory, on storage media, or on incoming files and also protect against other malwares.
  4. Install a personal **firewall** program to protect a computer and its data from unauthorized intrusions.
  5. Stay informed about new virus alerts and **virus hoaxes**. A **virus hoax** is an e-mail message that warns users of a non-existent virus or other malware and the contents of the hoax message, for example, may inform users that an important operating system file on their computer is a virus and encourage them to delete the file, which could make their computer unusable.
  6. Set the macro security in programs so that you can enable or disable macros and enable macros only if the document is from a trusted source and you are expecting it.
  7. Scan all downloaded programs for viruses and other malware.
  8. Scan any removable media for malware, before using the media.

b) Botnets

- A **botnet** is a group of compromised computers connected to a network that are used as part of a network that attacks other networks, usually for malicious purposes.
- **A zombie** is a compromised computer whose owner is unaware the computer is being controlled remotely by an outsider.
- A **bot** is a program that performs a repetitive task on a network. Cybercriminals install malicious bots on unprotected computers to create a botnet, also called a *zombie army.* The perpetrator then uses the botnet to send spam via e-mail, spread viruses and other malware, or commit a distributed denial of service attack

c) Denial of Service Attacks

- A **denial of service attack**, or **DoS attack**, is an assault whose purpose is to disrupt computer access to an Internet service such as the Web or e-mail.
- ***DDoS* (*distributed DoS*) *attack***, a more devastating type os DoS in which a zombie army is used to attack computers or computer networks.

d) Back Doors

- A **back door** is a program that allow users to bypass security controls when accessing a program, computer, or network. (*A rootkit can be a back door. Some worms leave back doors, which have been used to spread other worms or to distribute junk e-mail from the unsuspecting victim computers.)*

e) Spoofing

- **Spoofing** is a technique intruders use to make their network or Internet transmission appear legitimate to a victim computer or network.
  - *e-mail spoofing*, occurs when the sender's address or other components of the e-mail header are altered so that it appears the e-mail originated from a different sender, commonly is used for *virus hoaxes, spam, and phishing scams*.
  - *IP spoofing*, occurs when an intruder computer fools a network into believing its IP address is associated with a trusted source.

Safeguards against Botnets, DoS/DDoS Attacks, Back Doors, and Spoofing

- Some of the latest antivirus programs include provisions to protect a computer from DoS and DDoS attacks.
- Users can implement **firewall** solutions, **install intrusion detection software**, and set up **honeypots**.

1. **Firewalls**
   - **firewall** is hardware and/or software that protects a network's resources from intrusion by users on another network.

.

2. **Intrusion Detection Software**
   - software to identify possible security breaches, it automatically analyzes all network traffic, assesses system vulnerabilities, identifies any unauthorized intrusions, and notifies network administrators of suspicious behaviour patterns or system breaches.

3. **Honeypots**
   A *honeypot* is a vulnerable computer that is set up to entice an intruder to break into it so the organization can analyse an attack being perpetrated.

2. Unauthorized Access and Use

- **Unauthorized access** is the use of a computer or network without permission for unapproved or possibly illegal activities.
- Includes activities like: an employee using an organization's computer to send personal e-mail messages, an employee using the organization's word processing software to track his or her child's soccer league scores, or someone gaining access to a bank computer and performing an unauthorized transfer.

Safeguards against Unauthorized Access and Use

- have a written **acceptable use policy (AUP)** that outlines the computer activities for which the computer and network may and may not be used.
- use access controls to minimize the chances of perpetrators intentionally accessing or employees accidentally accessing confidential information on a computer.
- An ***access control*** is a security measure that defines *who* can access a computer, *when* and *what* actions they can take while accessing the computer.
- computer should maintain an **audit trail** that records in a file both successful and unsuccessful access attempts.
- **Identifying and Authenticating Users**
    - *Identification* verifies that an individual is a valid user.
    - *Authentication* verifies that the individual is the person he or she claims to be

Three methods of identification and authentication include user names and passwords, possessed objects, and bio metric devices.

a) User Names and Passwords

- A **user name**, or *user ID* (identification), is a unique combination of characters, such as letters of the alphabet or numbers, that identifies one specific user.
- A **password** is a private combination of characters associated with the user name that allows access to certain computer resources.
- some use *passphrases* to authenticate users instead of passwords.
- A **passphrase** is a private combination of words, often containing mixed capitalization and punctuation, associated with a user name that allows access to certain computer resources.
- Web sites use a CAPTCHA to further protect a user's password.
- A ***CAPTCHA***, Completely Automated Public Turing test to tell Computers and Humans Apart, is a program developed to verify that user input is not computer generated.

b) Possessed Objects

- A *possessed object* is any item that you must carry to gain access to a computer or computer facility e.g. badges, cards, smart cards, and keys.
- often are used in combination with personal identification numbers (PIN).
- A **personal identification number** (**PIN**) is a numeric password, either assigned by a company or selected by a user.

c) Biometric Devices

- **biometric device** authenticates a person's identity by translating personal characteristic, such as a fingerprint, into a digital code that is compared with a digital code stored in the computer.

3. Hardware Theft and Vandalism

- **Hardware theft** is the act of stealing computer equipment.
- **Hardware vandalism** is the act of defacing or destroying computer equipment.

Safeguards against Hardware Theft and Vandalism

- **Physical access controls**, such as locked doors and windows and install alarm systems for additional security. Some businesses use a *real time location system* (*RTLS*) to track and identify the location of high-risk or high-value items.

4. Software Theft

- **Software theft** is whereby someone
  1) steals software media,
  2) Intentionally erases programs,
  3) Illegally copies a program, or
  4) Illegally registers and/or activates a program.

Safeguards against Software Theft

- Keep original software boxes and media in a secure location, out of sight of prying eyes.
- Back up their files and disks regularly, in the event of theft.
- Software manufacturers issue users **license agreements** which is the right to use the software.
- **Product activation**, which is conducted either online or by telephone, users provide the software product's 25-character identification number to receive an installation identification number unique to the computer on which the software is installed.

5. Information Theft

- **Information theft** occurs when someone steals personal or confidential information.

Safeguards against Information Theft

- Implementing the user identification and authentication controls which are best suited for protecting information on computers located on an organization's premises.
- **Encryption,** is a process of converting readable data into unreadable characters to prevent unauthorized access. In the encryption process, the unencrypted, readable data is called *plaintext*. The encrypted (scrambled) data is called *ciphertext*.
  - An *encryption algorithm,* or *cypher,* is a set of steps that can convert readable plaintext into unreadable ciphertext.
  - An *encryption key* is a set of characters that the originator of the data uses to encrypt the plaintext and the recipient of the data uses to decrypt the ciphertext.
  - Two basic types of encryption are private key (*symmetric key encryption)* where both the originator and the recipient use the same secret key to encrypt and decrypt the data and public key (*asymmetric key encryption*) which uses two encryption keys: a public key and a private key.
- **Transport Layer Security** *Transport Layer Security* (*TLS)* provides encryption of all data that passes between a client and an Internet server.
- **Secure HTTP** *Secure HTTP* (*S-HTTP*) allows users to choose an encryption scheme for data that passes between a client and a server.

6. System Failure

- A *system failure* is the prolonged malfunction of a computer system which can cause loss of hardware, software, data, or information.
- Could be caused by aging hardware; natural disasters such as fires, floods, or hurricanes; random events such as electrical power problems; and even errors in computer programs.
  - **Electrical disturbances** - include noise, undervoltages, and overvoltages.
  - **Noise** is any unwanted signal, usually varying quickly, that is mixed with the normal voltage entering the computer.
  - An **undervoltage** occurs when the electrical supply drops.
  - An **overvoltage**, or **power surge**, occurs when the incoming electrical power increases and can cause immediate and permanent damage to hardware.

.

Safeguards against System Failure

- To protect against electrical power variations, use a **surge protector (*surge suppressor)***, uses special electrical components to smooth out minor noise, provide a stable current flow, and keep an overvoltage from reaching the computer and other electronic equipment
- An **uninterruptible power supply** (**UPS**) is a device that contains surge protection circuits and one or more batteries that can provide power during a temporary or permanent loss of power

Backing Up — The Ultimate Safeguard

- To protect against data loss caused by system failure or hardware, software, or information theft, computer users should back up files regularly.
- A **backup** is a duplicate of a file, program, or disk that can be used if the original is lost, damaged, or destroyed.

6. Describe How Data Is Kept Safe During Storage And Transmission

Data needs to be protected in three states: **at rest**, **in use**, and in **motion** as each state presents unique security challenges.

Data at Rest

Data at rest is data that is not actively moving from device to device or network to network such as data stored on a hard drive, laptop, flash drive, or archived/stored in some other way and is primarily protected by

1. Use of **firewalls** and **anti-virus** programs.
2. Encrypting hard drives or
3. storing individual data elements in separate locations to decrease the likelihood of attackers gaining enough information to commit fraud or other crimes.

Data in Use

Data in use is more vulnerable than data at rest because, by definition, it must be accessible to those who need it.

The keys to securing data in use are to

- **control access** as tightly as possible
- Use **authentication** to ensure that users aren't hiding behind stolen identities.
- Use audit trails to track and detect suspicious activity, diagnose potential threats, and proactively improve security.

Data in Motion

Data in transit, or data in motion, is data actively moving from one location to another such as across the internet or through a private network.

The best way to secure data in transmit is to use an **encryption** platform

BEST PRACTICES FOR DATA PROTECTION IN TRANSIT AND AT REST

Unprotected data, whether in transit or at rest, leaves enterprises vulnerable to attack, but there are effective security measures that offer robust data protection across endpoints and networks to protect data in both states. As mentioned above, one of the most effective data protection methods for both data in transit and data at rest is data encryption.

In addition to encryption, best practices for robust data protection for data in transit and data at rest include:

- Implement robust network security controls to help protect data in transit. Network security solutions like firewalls and network access control will help secure the networks used to transmit data against malware attacks or intrusions.
- Use proactive security measures that identify at-risk data and implement effective data protection for data in transit and at rest.
- Choose data protection solutions with policies that enable user prompting, blocking, or automatic encryption for sensitive data in transit, such as when files are attached to an email message or moved to cloud storage, removable drives, or transferred elsewhere.
- Create policies for systematically categorizing and classifying all company data, no matter where it resides, in order to ensure that the appropriate data protection measures are applied while data remains at rest and triggered when data classified as at-risk is accessed, used, or transferred.

7. Explore Techniques And Practices Of Risk Management

1. Ishikawa Diagram

The fishbone diagram, or the cause and effect diagram - allows breaking down a problem and identify the component parts.

2. Decision Tree

A decision tree is a diagram that branches in different directions with each "branch" highlighting one possible option, and can then branch off again further if the path splits again.

It is a visual way of plotting out risk management actions, especially if the options may result in vastly different solutions and costs.

3. Expert Interviews

Talk to the people who know, this helps with risk identification and also risk management action planning as the experts may offer different ways to address a risk, broadening options when it comes to deciding what to do next.

4. Workshops

Bringing subject experts together to help with risk management so as to a deeper insight into some of the possible threats (and opportunities) on the project.

5. SWOT Analysis

SWOT stands for Strength, Weakness, Opportunity and Threat. Helps identify the opportunities and threats of the project.

6. Risk Proximity Chart

Proximity refers to how close to the current date the risk is. For example, the risk of a supplier failing to deliver the requirement equipment can have a low proximity if the delivery date is far in the future. If the proposed delivery date is next week, the risk has a high proximity. Proximity can help to prioritize risks as it tells which ones are coming up.

7. Probability and Impact Matrix

It is a grid with probability on one side and impact on the other where each risk is rated against a standard scale. Allows tracking of risks over time and makes it easy to see the risks that require action.

8. Outline The Importance Of Securing Data At Off-Site Locations (Cloud Computing)

Off-site data protection, or **vaulting**,
Is the strategy of sending critical data out of the main location (*off* the main *site*) as part of a disaster recovery plan.

- This ensures systems and servers can be reloaded with the latest data in the event of a disaster, accidental error, or system crash.
- Also ensures that there is a copy of pertinent data that isn't stored on-site.

9. Identify Code Of Ethics And Professional Practices In The Computing Field

Computer ethics are the moral guidelines that govern the use of computers and information systems.

- areas of computer ethics are unauthorized use of computers and networks, software theft (piracy), information accuracy, intellectual property rights, codes of conduct, information privacy, and green computing.

1. Information Accuracy

Information accuracy today is a concern because many users access information maintained by other people or companies, such as on the Internet. Do not assume that because the information is on the Web that it is correct.

2. Intellectual Property Rights

- *Intellectual property* (*IP)* refers to unique and original works such as ideas, inventions, art, writings, processes, company and product names, and logos.
- These are the rights to which creators are entitled for their work.
- A **copyright** gives authors and artists exclusive rights to duplicate, publish, and sell their materials, it protects any tangible form of expression and ***piracy*** (making illegally copy) is infringement of copyright

3. Codes of Conduct

- An IT **code of conduct** is a written guideline that helps determine whether a specific computer action is ethical or unethical.

10. Discuss Security Policies, Laws And Computer Crime

11. Identify Relevant ICT Legislative And Regulatory Frameworks

Legislation to protect ***intellectual property rights***, include:

– Family Entertainment and Copyright Act of 2005
– U.S. Anticybersquatting Consumer Protection Act of 1999
– Digital Millennium Copyright Act (DMCA)

Privacy Laws

The concern about privacy has led to the enactment of federal and state laws regarding the storage and disclosure of personal data, and the laws stipulate that:

1. Information collected and stored about individuals should be limited to what is necessary to carry out the function of the business or government agency collecting the data.
2. Once collected, provisions should be made to restrict access to the data to those employees within the organization who need access to it to perform their job duties.
3. Personal information should be released outside the organization collecting the data only when the person has agreed to its disclosure.
4. When information is collected about an individual, the individual should know that the data is being collected and have the opportunity to determine the accuracy of the data.

12. Identify Environmental Impact From Computer Use And Disposal

Green computing involves reducing the electricity and environmental waste while using a computer and this include:
– The use of computers in an environmentally friendly manner
– Consumption of less Energy and paper

ENERGY STAR Program
• Developed to encourage the development of energy-saving devices

Energy Consumption and Conservation
• Power consumption and heat generation by computers are key concerns for businesses because more powerful computers use more energy and run hotter, increasing cooling costs

• *Some energy-saving features*
– Low-power sleep mode when not in use
– Energy- efficient flat-panel displays
– Liquid cooling systems
– CPUs that power up and down on demand
– Solar power and other alternatives

Solar systems Available for a number of applications
– Solar panels are built into the covers of some computer and tablet cases
– Portable solar panels can be attached to backpacks and other items

Hand-powered chargers can be used with portable computers, smartphones, and other mobile devices

Green Components
• Computers run quieter and cooler
• More recyclable hardware and packaging being used
• Amount of toxic chemicals in personal computers being reduced
• Recycled plastics being used in some mobile phones
• Built-in solar panels can charge devices

Recycling and Disposal of Computing Equipment
– Paper-based trash
• Paperless office basically a myth
• Almost one-half billion pieces of paper a year generated by printers worldwide
• Utilities designed to reduce paper consumption
– GreenPrint, PrintWhatYouLike.com
- Eliminate images, blank pages, non-critical content in order to print on the least amount of paper as possible

E-waste (e-trash)
• Discarded computer components
• Current hardware contain a variety of toxic and hazardous materials and Global concern is where it all eventually ends up

Proper recycling is essential
- Some recycling centres will accept computer equipment
– Many computer manufacturers have voluntary take-back programs
– Expired toner and ink cartridges can sometimes be returned to manufacturer or exchanged when purchasing new cartridges
– Using recharged printer cartridges saves consumers' money and helps reduce e-waste in landfills

13. Analyse The Ethical Impact Of Existing And Emerging Technologies

Employee monitoring involves the use of computers to observe, record, and review an employee's use of a computer, including communications such as e-mail messages, keyboard activity (used to measure productivity), and Web sites visited

Content filtering is the process of restricting access to certain material on the Web. Content filtering opponents argue that banning any materials violates constitutional guarantees of free speech and personal rights.

Social Engineering is defined as gaining unauthorized access or obtaining confidential information by taking advantage of the trusting human nature of some victims and the naivety of others.

14. Explain The Attributes Of Business Ethics

Ethics
– Overall standards of moral conduct
– Can vary with individual and religious beliefs, country, race, or culture

Personal Ethics
- Guide an individual's personal behaviour

Business Ethics
- Guide an individual's workplace behaviour

Computer Ethics
- Concern moral conduct related to computer use
- Individuals and businesses need to make ethical decisions every day

Ethical Use of Copyrighted Material
– Books and Web-Based Articles
• Need to properly credit sources to avoid plagiarism
• Plagiarism is a violation of copyright law and an unethical act
• Strict consequences for plagiarism at school and work
• Online tests for plagiarism are available and widely used by schools

Databases

1. Describe The File Based Approach Database Systems

- File based Approach is a traditional approach of collecting and storing data In which data is stored in files with each unit of organization having its own files and each file having a specific set of programs that manipulate the data in that file.

2. Outline Features Of A DBMS

DBMS

- Is a complex layer of software used to develop and to maintain the database and provides interface between the database and application programs.

Features of Database Management Systems

a) Data Dictionary Management

- **Data Dictionary**: contains names and descriptions of every data element in the database i.e. how data elements are related to each other.
- Stores data in a consistent memory, reducing data redundancy.
- **Data Languages**: a Special language (Data Definition Language) used to describe the characteristics of a data element placed into the Data Dictionary.

b) **Data Storage Management**

- DBMS Handles data storage in the database.
- Uses **Teleprocessing monitors** which is a software that manages communication between the database and remote terminals.

c) **Data Transformation and Presentation**

- DBMS provides options to transform data in multiple formats & then present it to the appropriate users.

d) **Security Management**

- DBMS provides security rules that can restrict access to tables, schema & other database objects.
- **Security Software**: provides tools used to shield the database from unauthorised access.

e) **Backup And Recovery Management**

- DBMS should be able to take backups and provide methods to recover the lost data.
- **Recovery and archiving system**: these allow data to be copied onto backups in case of disaster.

f) **Data Integrity Management**

- DBMS makes use of primary keys to make sure that data integrity is not affected in any case.
- The information in database management system occurs only once so chances of duplicity are very less, thus maintaining data integrity, data concurrency & data consistence

g) **Database Access Languages And Application Programming Interfaces**

- Use of Structured Query language to access database. The query language provided by DBMS is so easy to understand.
- **Report writers**: these are programs used to design output reports without writing an algorithm in any programming language.

3. Describe Features Of Relational Database Which Address Limitations Of A File Based Approach

a) **Program-Data Independence**

- Data is independent of the programs that accesses it, in the file-based system, the structure of the data files is defined in the application programs so if a user wants to change the structure of a file, all the programs that access that file might need to be changed as well.
- Relational databases is *self-describing, ie.* Contains the database itself and also *metadata* which defines and describes the data and relationships between tables in the database (DD).
- Therefore, one change is all that is needed to change the structure of a file.

b) **Support for multiple views of data**

- A database supports multiple views of data, ie. Subsets of the database, which is defined and dedicated for particular users of the system.
- Multiple users in the system might have different views of the system and each view might contain only the data of interest to a user or group of users.

c) **Sharing of data and multiuser system**

- allow many users to access the same database at the same time through use of c**oncurrency control strategies (***features of a database that allow several users access to the same data item at the same time*)
- These strategies ensure that the data accessed are always correct and that data **integrity** is maintained.

d) **Control of data redundancy/ Eliminating data redundancy**

- Data need only be stored once and applications that need data can access the data from the central database.

e) **Data sharing**

- allows for data sharing among employees and others who have access to the system.
- gives users the ability to generate more information from a given amount of data than would be possible without the integration of all the data, for an organization, within a database system.

f) **Enforcement of integrity constraints**

- define and enforce certain constraints to ensure that users enter valid information and maintain data integrity.
- A *database constraint* is a restriction or rule that dictates what can be entered or edited in a table
- Database constraints like *Data uniqueness* ensures that no duplicates are entered thus ensuring consistence and integrity.

g) **Restriction of unauthorized access**

- Database management system provide a security system to create and control different types of user accounts and restrict unauthorized access.

- Users of a database system are given different access privileges e.g ***read-only*** *access* (i.e., the ability to read a file but not make changes), or ***read and write*** *privileges*, which is the ability to both read and modify a file.
- Databases implement security through ***data classification*** (i.e., data objects are given classification levels and users are assigned clearance levels) and ***data encryption***

h) **Data independence**
- The system data descriptions or data describing data (metadata) are separated from the application programs.
- The database approach allows multiple application programs to use the data concurrently and the data can be accessed in several different ways (e.g., through applications processing, online query, and report writing programs).
- The access can be quickly changed by modifying the definition of the tables or views.

i) **Transaction processing**
- Database management system include concurrency control subsystems which ensures that data remains consistent and valid during transaction processing even if several users update the same information.

j) **Backup and recovery facilities**
- Database system provides for backing up and recovering of data.
- If a computer system fails in the middle of a complex update process, the recovery subsystem is responsible for making sure that the database is restored to its original state.

k) **Ease of maintenance.**
- Because each data element is stored only once, any additions, deletions, or changes to the database are accomplished easily.

l) **Reduced storage costs.**
- By eliminating redundant data, storage space is reduced, resulting in associated cost savings.

Relational Database Vs Flat File

Relational database	**Flat file**
Less duplication of data as data does not need to in every table	Too much duplication of data since every table repeats data
Offer greater data integrity as there is little chance of getting duplicate of data	No data integrity is guaranteed due to too much duplication of data
Data is available to all users of the system as there are no problems of software incompatibility. Thus users share files	Data not available everywhere as there is no sharing of files
Creates different user views within the DBMS	No different user views are created
Easy to access data from different table due to relationships between the tables	Difficult to access data from different files as the files are not linked
Retrieval of records from different files is faster	Retrieval of records from different files is slower
Better security of records is enhanced	Less security of data from unauthorised access
Promote program data independence	There is program data dependence
There is centralised management of data which is more efficient	No central data management, difficult to manage and less security

4. Normalise Database Tables

Database normalisation

- is the process of organising fields and tables of a relational database to minimize redundancy and dependency.
- is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

Data in a table is normalised to a specific form to prevent possible occurrence of update anomalies, basing on functional dependencies among the attributes in the relation. It is done so as:

- To reduce duplication of data by storing each record/data within the database only once
- To ensure that data stored in databases is consistent.
- Produce controlled redundancies to link tables
- To put data into a more flexible form that is able to accurately accommodate change
- To avoid certain anomalies like updating, insertion and deleting anomalies
- To allow users to make queries relating to data stored in different tables

Normalization

1NF - First normal form

- A table is in 1NF if and only if an attribute (column) of a table do not hold multiple values i.e it should have only atomic values.
- All entries in a field must be of same kind and each field must have a unique name, but the order of the field (column) is irrelevant.

Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123 8123450987

Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field. Therefore this table is **not in 1NF** as “each attribute of a table must have atomic (single) values”, the emp_mobile values for employees Jon & Lester violates that rule.

To make the table complies with 1NF we should have the data like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

2NF - Second normal form

- The table in 1NF will lead to too much duplication of data, therefore a table will be in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key i.e. there are no **partial dependencies**.
- A table is in 2NF if is in 1NF (First normal form) and no non-prime attribute is dependent on the proper subset of any candidate key of table.

****NB:** An *attribute* that is not part of any candidate/primary key is known as non-prime/ non-key attribute.

Example: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys: {teacher_id, subject}
Non prime attribute: teacher_age

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone.

To put the table in 2NF we can break it in two tables like this:

teacher_details table:

teacher_id	teacher_age
111	38
222	38
333	40

teacher_subject table:

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

3NF - Third normal form

- A table is in *Third Normal Form (3NF)* if it is in 2NF and has no transitive dependencies. A *transitive dependency* is when an attribute is indirectly functionally dependent on the key (the dependency is through another nonkey attribute), that is, all non-key elements are fully dependent on the primary key.
- In other words a table is in 3NF if it is in 2NF and for each functional dependency X-> Y , X is a super key of table and Y is a prime attribute (is a part of one of the candidate keys) of table

Example: Suppose a company wants to store the complete address of each employee, they create a table named **employee_details** that looks like this:

emp_id	**emp_name**	**emp_zip**	**emp_state**	**emp_city**	**emp_district**
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

Super keys: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…so on
Candidate Keys: {emp_id}
Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.

Here, emp_state, emp_city & emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id). Therefore to put the table in 3NF we have to break the table into two tables to remove the transitive dependency:

employee table:

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

employee_zip table:

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

5. Design Database Applications Using Entity Relationship Diagrams (ERDs)

An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities (which will become your tables) and their relationships to each other.

Entity: physical objects like person, patient or events on which information or data is being collected. It can also be an abstract object like a patient record.
- An *entity* is something of interest to an organisation about which data is to be held.
- represents a table in a database

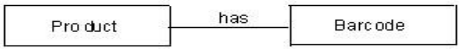
Attribute: individual data item within an entity; e.g. date of birth, surname.
- This is a property or characteristic of an entity.
- An attribute or a column (simple) represents a piece of data in the table, like an address, salary, and date.

Relationship: links between two different entities or relations (tables).
- a link or association between entities.

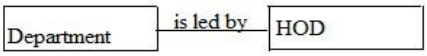
Types of Relationship

- **One-to-one**

  Eg Products in a supermarket each have a unique barcode number.

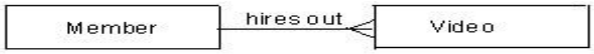


A department in school is led up by a HOD, and this person only leads one department



- **One-to-many**

  Eg A video club member may hire out a number of videos.

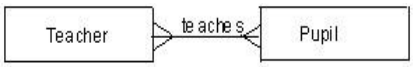


The head of department may be in charge of many staff, but these staff members only have one head of department.

- **Many to One**

  Many videos can be hired by one member.

  Teachers and pupils in a school. Each teacher teaches many pupils and each pupil has many teachers.



A teacher may order many books, but each book could be ordered by many teachers.

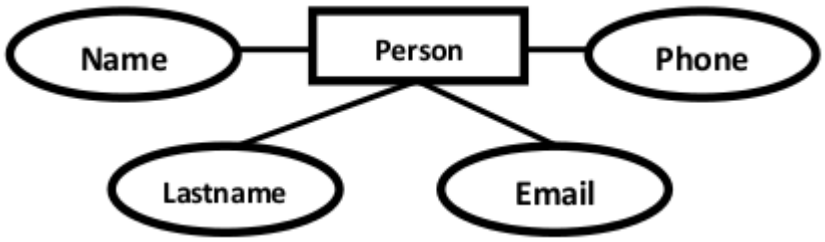
6. Convert ERDs to Relational Databases Schema in Standard Normal Form

ER diagram represents the conceptual level of database design meanwhile the relational schema is the logical level for the database design. We will be following the simple rules:

1. Entities and Simple Attributes:

- An *entity* within ER diagram is turned into a **table**. You may preferably keep the same name for the entity or give it a sensible name but avoid DBMS reserved words as well as avoid the use of special characters.
- Each *attribute* turns into a **column** (attribute) in the table. The key attribute of the entity is the **primary key** of the table which is usually underlined. It can be composite if required but can never be null.

Taking the following simple ER diagram:



The initial relational schema is expressed in the following format writing the table names with the attributes list inside a parentheses as shown below for

Persons (personId , name, lastname, email)

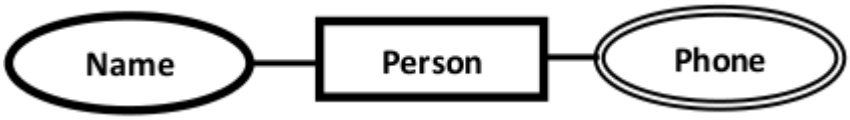
Persons Table

personID	name	lastname	email

- personId is the primary key for the table Persons

2. **Multi-Valued Attributes**

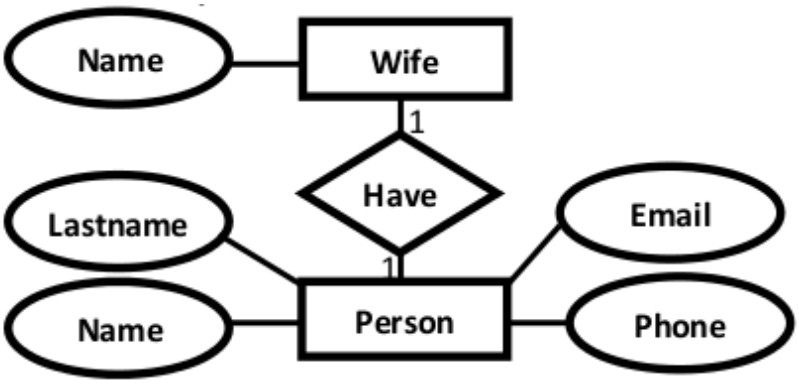
A multi-valued attribute is usually represented with a double-line oval.



- A multi-valued attribute is turned into a new **entity** or **table** of its own.
- make a 1:N relationship between the new entity and the existing one ie
  - Create a table for the attribute.
  - Add the primary (id) column of the parent entity as a foreign key within the new table as shown below:

Persons(personid , name, lastname, email)
Phones (phoneid , ***personid***, phone)
****NB:* personid** within the table Phones is a *foreign key* referring to the personid of Persons

3. 1:1 Relationships

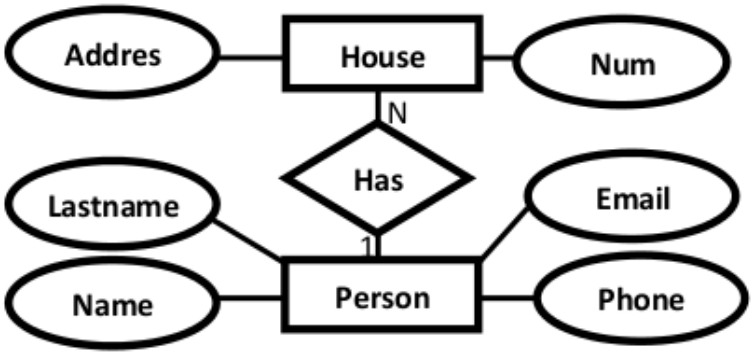


For instance, let us consider the case where the Person has or optionally has one wife. You can place the primary key of the wife within the table of the Persons which we call in this case Foreign key as shown below.

Persons(personid , name, lastname, email , ***wifeid***)
Wife (wifeid , name)
Or vice versa to put the **personid** as a foreign key within the Wife table as shown below:
Persons(personid , name, lastname, email)
Wife (wifeid , name , ***personid***)

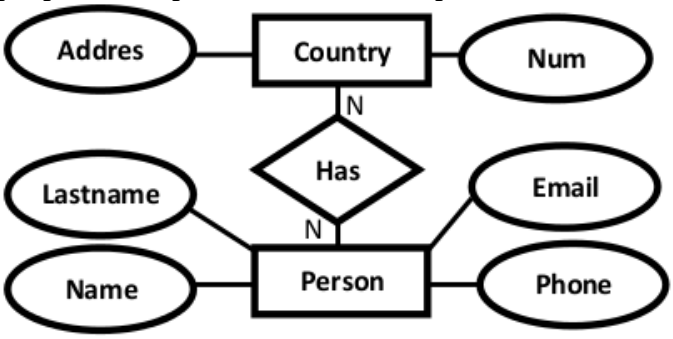
4. 1:N Relationships (One to Many)

For instance, the Person can have a **House** from zero to *many* , but a **House** can have only one **Person**. To represent such relationship the **personid** as the Parent node must be placed within the Child table as a foreign key but **not the other way around.**



It should convert to :
Persons(personid , name, lastname, email)
House (houseid , num , address, ***personid***)

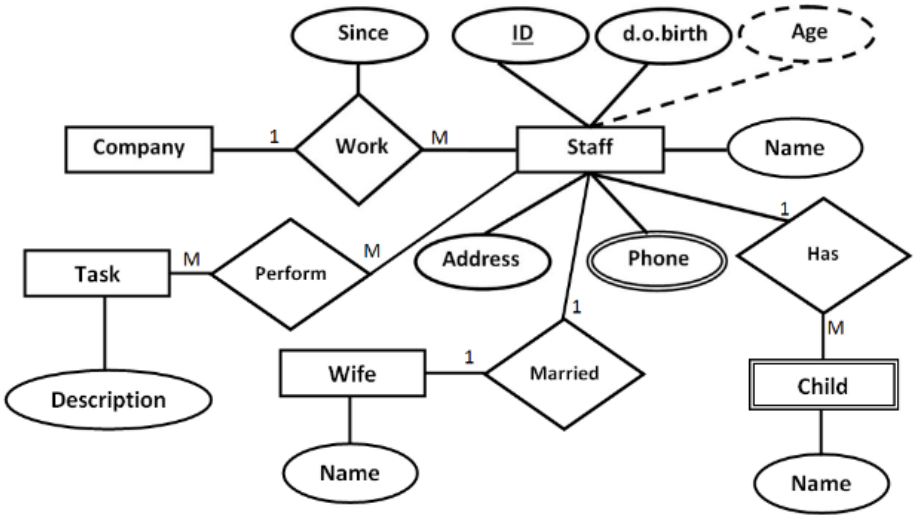
5. N:N Relationships (Many to Many)
For instance, The Person can live or work in many countries. Also, a country can have many people. To express this relationship within a relational schema we use a separate table.



It converte into :

Persons(personid , name, lastname, email)
Countries (countryid , name, code)
HasRelat (hasrelatid , **personid , countryid**)
** *It is recommended to use table to represent relationship with attributes to keep the design tidy and clean **regardless of the cardinality** of the relationship*.

Case Study
Produce the relational schema for the following ER diagram:



The relational schema for the ER Diagram is given below as:

Company(CompanyID , name , address)
Staff(StaffID , dob , address , ***WifeID***)
Child(ChildID , name , ***StaffID***)
Wife (WifeID , name)
Phone(PhoneID , phoneNumber , ***StaffID***)
Task (TaskID , description)
Work(WorkID , ***CompanyID , StaffID*** , since)
Perform(PerformID , ***StaffID , TaskID***)

7. Write Sql Commands

- SQL is the standard computer language used to communicate with relational database management systems.
- SQL commands consist of English-like statements which are used to *query, insert, update, and delete* data.
- SQL is a nonprocedural database language because when retrieving data from the database it is enough to tell SQL what data to be retrieved, rather than how to retrieve it.

A **query** is a user request to retrieve data or information using a certain condition.
It is a command written in query language which allows users to access and manipulate data stored in databases.

These SQL commands are mainly categorized into four categories:

1. **DDL(Data Definition Language) :** consists of the SQL commands that can be used to define the database schema. It deals with descriptions of the database schema and is used to create and modify the structure of database objects in database.
   Examples of DDL commands:
   - **CREATE** – used to create the database or its objects (like table, index, function, views, store procedure and triggers).
   - **DROP** – used to delete objects from the database.
   - **ALTER**- used to alter the structure of the database.
   - **TRUNCATE**– used to remove all records from a table, including all spaces allocated for the records are removed.
   - **COMMENT** – used to add comments to the data dictionary.
   - **RENAME** – used to rename an object existing in the database.
2. **DML(Data Manipulation Language) :** deals with the manipulation of data in database.
   Examples of DML:
   - **SELECT** – used to retrieve data from the a database.
   - **INSERT** – used to insert data into a table.
   - **UPDATE** – used to update existing data within a table.
   - **DELETE** – used to delete records from a database table.
3. **DCL(Data Control Language) :** DCL includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.
   Examples of DCL commands:
   - **GRANT**-gives user's access privileges to database.
   - **REVOKE**-withdraw user's access privileges given by using the GRANT command.

4. **TCL(transaction Control Language) :** TCL commands deals with the transaction within the database.
   Examples of TCL commands:
   - **COMMIT**– commits a Transaction.
   - **ROLLBACK**– rollbacks a transaction in case of any error occurs.
   - **SAVEPOINT**– sets a save point within a transaction.
   - **SET TRANSACTION**–specify characteristics for the transaction.

CREATE Query

There are two CREATE statements available in SQL, i.e. CREATE DATABASE and CREATE TABLE

CREATE DATABASE
The **CREATE DATABASE** statement is used to create a new database in SQL.
Syntax:

```
CREATE DATABASE database_name;

database_name: name of the database.
```

ExampleQuery:
This query will create a new database in SQL and name the database as *my_database*.

```
CREATE DATABASE my_database;
```

CREATE TABLE
The CREATE TABLE statement is used to create a table in SQL

Since a table comprises of rows and columns will have to provide all the information to SQL about the names of the columns, type of data to be stored in columns, size of the data etc.

Syntax:

```
CREATE TABLE table_name

(

column1 data_type(size),

column2 data_type(size),

column3 data_type(size),

....

);

table_name:  name of the table.
column1 name of the first column.
data_type: Type of data we want to store in the particular column e.g.int for integer data.
size: Size of the data we can store in a particular column.
```

SELECT Query

- The SELECT Statement in SQL is used to retrieve or fetch data from a database.
- The SELECT clause of a SELECT command statement, specify the columns that are to be displayed in the query result and, optionally, which column headings to be above the result table.

SampleTable:

Student

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	Delhi	9156768971	18

Basic Syntax:

```
SELECT column1,column2 FROM table_name
column1 , column2: names of the fields of the table
table_name: from where we want to fetch
```

- To fetch the entire table or all the fields in the table:

```
SELECT * FROM table_name;
```

- Query to fetch the fields ROLL_NO, NAME, AGE from the table Student:

```
SELECT ROLL_NO, NAME, AGE FROM Student;
```

Output:

ROLL_NO	NAME	Age
1	Ram	18
2	RAMESH	18
3	SUJIT	20
4	SURESH	18

- To fetch all the fields from the table Student:

```
SELECT * FROM Student;
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	9455123451	18

ROLL_NO	NAME	ADDRESS	PHONE	Age
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	Delhi	9156768971	18

INSERT INTO Statement

The INSERT INTO statement of SQL is used to insert a new row in a table. There are two ways of using INSERT INTO statement for inserting rows:

1. **Only values:** First method is to specify only the value of data to be inserted without the column names.
   Syntax:

```
INSERT INTO table_name VALUES (value1, value2, value3,...);
```

table_name*: name of the table.*
value1, value2,.. *: value of first column, second column,... for the new record*

2. **Column names and values both:** In the second method we will specify both the columns which we want to fill and their corresponding values:
   Syntax:

```
INSERT INTO table_name (column1, column2, column3,..) VALUES (value1, value2, value3,..);
```

table_name*: name of the table.*
column1*: name of first column, second column ...*
value1, value2, value3 *: value of first column, second column,... for the new record*

Student

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	Delhi	9156768971	18
3	SUJIT	ROHTAK	9156253131	20
2	RAMESH	GURGAON	9652431543	18

Queries:

Method 1 (Inserting only values) :

```
INSERT INTO Student VALUES ('5','HARSH','WEST BENGAL','8759770477','19');
```

Output:
The table **Student** will now look like:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	9455123451	18
2	RAMESH	GURGAON	9562431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	Delhi	9156768971	18
3	SUJIT	ROHTAK	9156253131	20
2	RAMESH	GURGAON	9562431543	18
5	HARSH	WEST BENGAL	8759770477	19

Method 2 (Inserting values in only specified columns):

```
INSERT INTO Student (ROLL_NO, NAME, Age) VALUES ('5','PRATIK','19');
```

Output:
The table **Student** will now look like:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	9455123451	18
2	RAMESH	GURGAON	9562431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	Delhi	9156768971	18
3	SUJIT	ROHTAK	9156253131	20
2	RAMESH	GURGAON	9562431543	18
5	PRATIK	null	null	19

Notice that the columns for which the values are not provided are filled by null. Which is the default values for those columns.

Using SELECT in INSERT INTO Statement

We can use the SELECT statement with INSERT INTO statement to copy rows from one table and insert them into another table.The use of this statement is similar to that of INSERT INTO statement. The difference is that the SELECT statement is used here to select data from a different table.

Inserting all columns of a table: We can copy all the data of a table and insert into in a different table.

Syntax:

```
INSERT INTO first_table SELECT * FROM second_table;
```

first_table*: name of first table.*
second_table*: name of second table.*

We have used the SELECT statement to copy the data from one table and INSERT INTO statement to insert in a different table.

- **Inserting specific columns of a table:** We can copy only those columns of a table which we want to insert into in a different table.

  Syntax:

  ```
  INSERT INTO first_table(names_of_columns1) SELECT names_of_columns2
  FROM second_table;
  ```

  first_table*: name of first table.*
  second_table*: name of second table.*
  names of columns1*: name of columns separated by comma(,) for table 1.*
  names of columns2*: name of columns separated by comma(,) for table 2.*

  We have used the SELECT statement to copy the data of the selected columns only from the second table and INSERT INTO statement to insert in first table.

- **Copying specific rows from a table**: We can copy specific rows from a table to insert into another table by using WHERE clause with the SELECT statement. We have to provide appropriate condition in the WHERE clause to select specific rows.

  Syntax:

  ```
  INSERT INTO table1 SELECT * FROM table2 WHERE condition;
  ```

  first_table*: name of first table.*
  second_table*: name of second table.*
  condition: *condition to select specific rows.*

UPDATE Statement

The UPDATE statement in SQL is used to update the data of an existing table in database. We can update single columns as well as multiple columns using UPDATE statement as per our requirement.

Basic Syntax

```
UPDATE table_name SET column1 = value1, column2 = value2,...
WHERE condition;
```

table_name: *name of the table*
column1*: name of first , second, third column....*
value1*: new value for first, second, third column....*
condition*: condition to select the rows for which the values of columns needs to be updated.*

NOTE: In the above query the **SET** statement is used to set new values to the particular column and the **WHERE** clause is used to select the rows for which the columns are needed to be updated.

Student

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	Delhi	9156768971	18
3	SUJIT	ROHTAK	9156253131	20
2	RAMESH	GURGAON	9652431543	18

Example Queries

- **Updating single column**: Update the column NAME and set the value to 'PRATIK' in all the rows where Age is 20.

```
UPDATE Student SET NAME = 'PRATIK' WHERE Age = 20;
```

Output:

This query will update two rows(third row and fifth row) and the table **Student** will now look like,

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	9455123451	18
2	RAMESH	GURGAON	9562431543	18
3	PRATIK	ROHTAK	9156253131	20
4	SURESH	Delhi	9156768971	18
3	PRATIK	ROHTAK	9156253131	20
2	RAMESH	GURGAON	9562431543	18

- **Updating multiple columns:** Update the columns NAME to 'PRATIK' and ADDRESS to 'SIKKIM' where ROLL_NO is 1.

```
UPDATE Student SET NAME = 'PRATIK', ADDRESS = 'SIKKIM' WHERE ROLL_NO = 1;
```

Output:
The above query will update two columns in the first row and the table **Student** will now look like,

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	PRATIK	SIKKIM	9455123451	18
2	RAMESH	GURGAON	9562431543	18
3	PRATIK	ROHTAK	9156253131	20
4	SURESH	Delhi	9156768971	18
3	PRATIK	ROHTAK	9156253131	20
2	RAMESH	GURGAON	9562431543	18

Note: For updating multiple columns we have used comma(,) to separate the names and values of two columns.

- **Omitting WHERE clause:** If we omit the WHERE clause from the update query then all of the rows will get updated.

```
UPDATE Student SET NAME = 'PRATIK';
```

Output:
The table **Student** will now look like,

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	PRATIK	Delhi	9455123451	18
2	PRATIK	GURGAON	9562431543	18
3	PRATIK	ROHTAK	9156253131	20
4	PRATIK	Delhi	9156768971	18
3	PRATIK	ROHTAK	9156253131	20
2	PRATIK	GURGAON	9562431543	18

DELETE Statement

The DELETE Statement in SQL is used to delete existing records from a table. We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.

Basic Syntax:

```
DELETE FROM table_name WHERE some_condition;
```

table_name: name of the table
some_condition: condition to choose particular record.

Note: We can delete single as well as multiple records depending on the condition we provide in WHERE clause. If we omit the WHERE clause then all of the records will be deleted and the table will be empty.

Sample Table:

Student

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	Delhi	9156768971	18
3	SUJIT	ROHTAK	9156253131	20
2	RAMESH	GURGAON	9652431543	18

Example Queries:

Deleting single record: Delete the rows where NAME = 'Ram'. This will delete only the first row.

```
DELETE FROM Student WHERE NAME = 'Ram';
```

Output:

The above query will delete only the first row and the table **Student** will now look like,

ROLL_NO	NAME	ADDRESS	PHONE	Age
2	RAMESH	GURGAON	9562431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	Delhi	9156768971	18
3	SUJIT	ROHTAK	9156253131	20
2	RAMESH	GURGAON	9562431543	18

Deleting multiple records: Delete the rows from the table Student where Age is 20. This will delete 2 rows(third row and fifth row).

```
DELETE FROM Student WHERE Age = 20;
```

Output:

The above query will delete two rows(third row and fifth row) and the table **Student** will now look like,

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	9455123451	18
2	RAMESH	GURGAON	9562431543	18
4	SURESH	Delhi	9156768971	18
2	RAMESH	GURGAON	9562431543	18

Delete all of the records: There are two queries to do this as shown below,

```
query1: "DELETE FROM Student";


query2: "DELETE * FROM Student";
```

Output:

All of the records in the table will be deleted, there are no records left to display. The table **Student** will become empty!

8. Develop Interfaces and Queries Using DBMS Tools

In **Microsoft Access**,

tblExams

STUDENT NUM	AMOUNT PA	DATE PAID	NO OF SUBJECTS	RECEIPT NUI
0001/	$25.00	7/4/2010	10	0001
0002/	$15.00	5/2/2009	9	0002
0009/	$25.00	4/4/2010	8	0007
0007/	$24.00	7/3/2010	5	0034
0008/	$13.00	4/3/2010	7	0045
0005/	$24.00	4/1/2010	9	0046
0010/	$16.00	7/4/2010	3	0049
0006/	$36.00	3/5/2010	10	0456
0004/	$27.00	7/3/2010	8	0894
0003/	$40.00	5/2/2009	8	1455
*	$0.00		0	

If one wants to search students who paid $24 and the number of Subjects as 5, he enters the following in the design view of the table query;

Field:	[STUDENT NUMBER]	[AMOUNT PAID]	[DATE PAID]	[NO OF SUBJECTS]	[RECEIPT NUMBER]
Table:	tblExams	tblExams	tblExams	tblExams	tblExams
Sort:					
Show:	☑	☑	☑	☑	☑
Criteria:		24		5	
or:					

The above can be written in SQL as given below in order to produce the same result:

SELECT [ALL / DISTINCT] *expr1* [AS *col1*], *expr2* [AS *col2*] ;
FROM *tablename* WHERE *condition*

SELECT *tblExams.[STUDENT NUMBER], tblExams.[AMOUNT PAID], tblExams.[DATE PAID], tblExams.[NO OF SUBJECTS], tblExams.[RECEIPT NUMBER]*
FROM *tblExams*
WHERE *(((tblExams.[AMOUNT PAID])=25)* **AND** *((tblExams.[NO OF SUBJECTS])=5));*

The SQL will produce the following result:

tblExams Query

STUDENT NUM	AMOUNT PA	DATE PAID	NO OF SUBJECTS	RECEIPT NUI
0007/	$24.00	7/3/2010	5	0034
*	$0.00		0	

SELECT * FROM *tblExams*;
Here the asterisk symbol indicates the selection of all the columns of the table.

SELECT *name, mtest* **FROM** *student* ;
WHERE *class*="1A" **AND** ;
mtest **BETWEEN** *80* **AND** *90*

9. Access Data in A Database Through A High Level Language

Dbase Connection in VB 6.0

```
Dim cn as adodb.connection
Private Sub Form_Load()
Dim connstring As String
  Set cn = New ADODB.Connection
  connstring = “Provider=Microsoft.Jet.OLEDB.4.0;
  Data Source=c:\newadodb\devin.mdb"
  cn.Open connstring
End Sub
```

Add Records to Dbase

```
Private Sub cmdadd_Click()
   dim rs as adodb.recordset
   set rs = new adodb.recordset
   Dim strsql As String
   strsql = "select * from login where firstname = null"
   rs.Open strsql, cn, adOpenKeyset, adLockOptimistic
   rs.AddNew
   rs.Fields("id") = Val(Txtid.text)
   rs.Fields("firstname") = txtfirstname.text
   rs.Fields("lastname") = txtlastname.text
   rs.Fields("age") = Val(txtage.text)
   rs.Update
   rs.Close
End Sub
```

Update Records in Dbase

```
Private Sub cmdupdate_Click()
Dim rs as adodb.recordset            ‘declare recordset
Set rs = new adodb.recordset         ‘set the recordset
Dim strsql As String
strsql = “select * from login where ID = “ & txtid.text & “”
rs.Open strsql, cn, adOpenKeyset, adLockOptimistic
If rs.eof then   ‘Record Does Not Exist
  Msgbox “Record Not Found”
Else
  rs.Fields("age") = Val(txtage.text)
  rs.Update
End if
rs.close
End Sub
```

Algorithm Design and Data Structures

1. Distinguish Between Dynamic and Static Data Structures

Data structure is a collection of different data items that are stored together as a single unit and the operations allowable on them.

- It is the way in which data is organised or grouped into a single unit

Dynamic Data Structures

- increase and decrease in size while the program is running, e.g. binary trees, linked lists, etc.
- uses space needed at a particular time, no limitations at all, unless computer memory is full

Advantages of Dynamic Data Structures

- Only uses the space that is needed at any time
- Makes efficient use of the memory, no spaces lie idle at any given point
- Storage no-longer required can be returned to the system for other uses.
- Does not allow overflow
- There is effective use of resources as resources are allocated at run-time, as they are required.

Disadvantages of Dynamic Data Structures

- Complex and more difficult to program as software needs to keep track of its size and data item locations at all times
- Can be slow to implement searches
- A linked list only allows serial access

Static data structure

- do not change in size while the program is running, e.g **arrays** and **fixed length records**.
- Most arrays are static, i.e., once you declare them, they cannot change in size.
- It main advantage is that amount of storage is known and therefore is easier to program

Advantages of Static Data Structures

- Easy to check for overflow
- Memory allocation is fixed and therefore there is no problem with adding and removing data items.
- Compiler can allocate space during compilation
- locations of each element is fixed and known by the program.
- Easy to program as there is no need to check for data structure size at any given time/point
- An array allows random access and is faster to access elements than in other data structures

Disadvantages of Static Data Structures

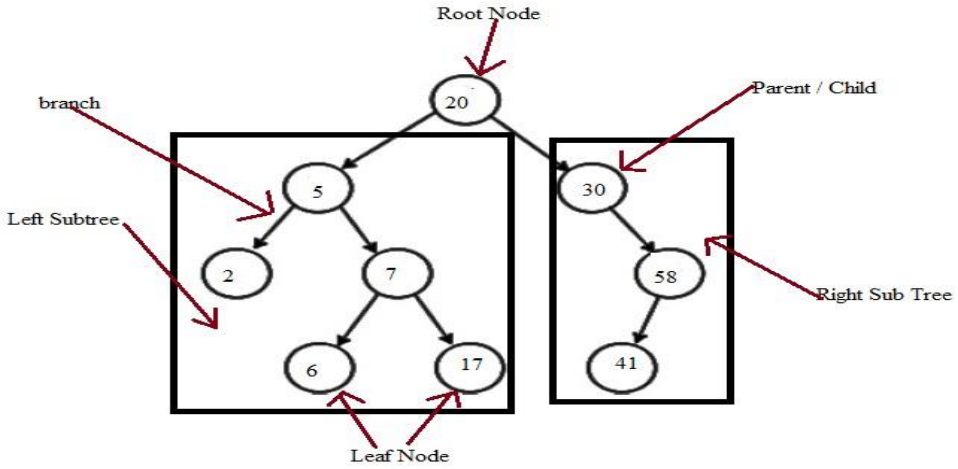
- Programmer has to estimate maximum amount of space needed, which may be difficult
- Can waste a lot of space if some space is left with no data entered into it.

*** In **Static data structure** the size of the **structure** is fixed i.e. the **data structure** can be modified but without changing the memory space allocated to it. In **Dynamic data structure** the size of the **structure** in not fixed and can be modified during the operations performed on it.*

2. Perform Operations On Binary Trees and Arrays

Binary Trees

A binary tree is a data structure, consisting of a root node and zero, one or two sub-tree which are organised in a hierarchical way. Each node is a parent of at most two nodes.



Constructing a Binary Tree

Place first item into root node
For subsequent items, start from root node
Repeat
If *(new item > this node item)* ***Then***
Follow right pointer
Else
Follow left pointer
Endif
Until *pointer =0*
Place item at this node

Binary tree traversal

Tree traversal means walking through the trees' structure such that each node is visited once. Common traversal methods are: *Pre-Order, In-order and Post-Order Traversals.*

Pre-Order traversal

The order of traversal is:

- Visit the ***Root*** Node
- Traverse the ***Left*** sub-tree
- Traverse the ***Right*** sub-tree.

In-Order Traversal

The order of traversal is:

- Traverse the ***Left*** sub-tree
- Visit the ***Root*** node
- Traverse the ***Right*** sub-tree.

NB*: In-order traversal prints items in* ***ascending order*** *or in* ***alphabetical order*** *In-order traversal*

Post-Order Traversal

The order of traversal is:

- Traverse the ***Left*** sub-tree
- Traverse the ***Right*** sub-tree.
- Visit the ***Root*** node

Inserting Data into a Binary Tree

- *Look at each node starting from the root node*
- *If the root is empty, create the node and place the value*
- *If the new value is less than the value of the of the node, move **left**, else move right*
- *Repeat this for each node arrived until there is no node*
- *Then create a new node and insert the data.*
- *Perform until no new item need to be added*

This can be written algorithmically as:

```
Repeat Compare new value with root value
    If new value > root value then
        Follow right sub-tree
    Else
        Follow left sub-tree
    Endif
Until no sub-tree
Insert new value as root of new sub-tree.
```

Deleting Data from a Tree

Data in a tree serves two purposes (one is to be the data itself) the other is to act as a reference for the creation of the subtree below it therefore Deleting data from a tree is quite complicated, because if it has sub-nodes, these will also be deleted.

There are two options however:

1. The structure could be **left the same**, but the value of that node **set to deleted**.
2. The tree could be traversed, the **values removed**, put into a stack and then put back into a binary tree.

Problems when deleting element from tree and solution

- The value at the node is not only data, it is part of the structure of the tree
- If the node is simply deleted, then the sub-tree leading from it is not navigable

Possible scenarios for deleting a Node

1. Node is a **leaf node**: this is straight forward as it does not affect the tree structure.
2. Node is having **one child**: Establish a direct link between parent node and child node of this node.
3. Node has ***two children*** (Root Node): This is little tricky.. the steps involved in this are.
    1. First find the successor (or predecessor) of the this node.
    2. Delete the successor (or predecessor) from the tree.
    3. Replace the node to be deleted with the successor (or predecessor)

Successor: In a binary tree successor of a node is the smallest node of the tree that is strictly greater than this node.

There are two possible cases with a node (successor or predecessor)

- A node has right children: In this case the leftmost child of the right sub-tree will be successor.
- A node has no children: Go to the parent node and find a node for which this node is part of left sub-tree.

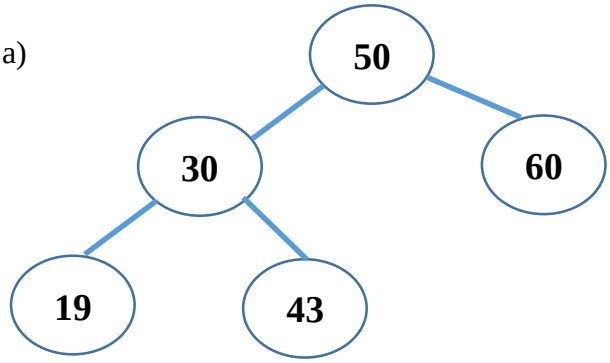
Example

a) Produce a binary tree for the following array
50,60,30,43,19

b) Assuming that the element 50 is completely deleted and removed from the tree structure, draw a new binary tree structure after 50 has been removed

solution

a)



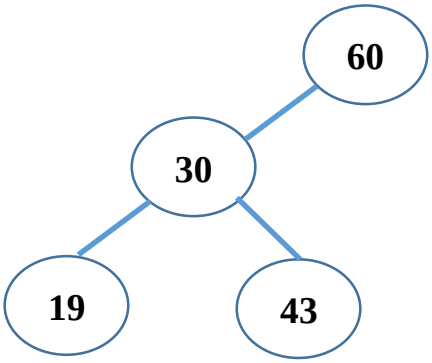
b) The node to be deleted has two child nodes, therefore

- Firstly find the successor of this node – to do this use the in-order traversal method to traverse the tree

⇨ 19, 30, 43, 50, 60 } *output obtained using in-order*

This is the successor of 50

- Now delete the **successor** (**60**) from the tree and
- Replace the node to be deleted (50) with the successor (60)



Searching item from Binary Tree

The algorithm is as follows:

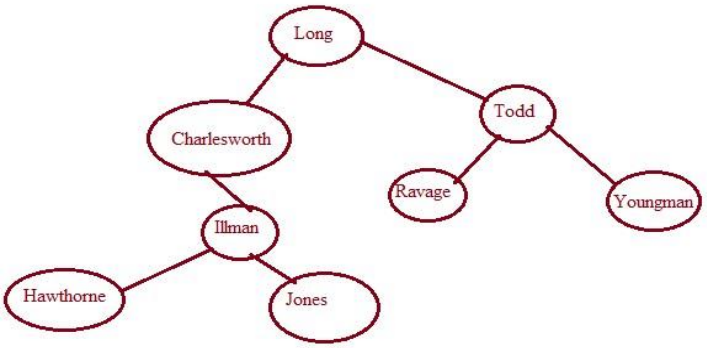
```
Enter item to search(this item) Start at root
node
 Repeat
        If wanted item = this item Then
            Found = True
            Display Item
    Else
            If wanted Item >this item Then
                Follow right pointer
            Else
                Follow left pointer
            EndIf
    EndIf
  Until (Found=True or Null pointer is encountered)
```

IMPLEMENTATION OF BINARY TREES USING ARRAYS

Binary trees can be implemented using left and right pointers for each node. Each node will have the following:

- Left pointer
- Right pointer
- Data item

Let's take the binary tree below for the list of elements:
Long, Charlesworth, Illman, Hawthon, Todd, Youngman, Jones, Ravage.



These can be illustrated as follows:

	Left	Data	Right
Tree [1]	2	Long	5
Tree [2]	0	Charlesworth	3
Tree [3]	4	Illman	7
Tree [4]	0	Hawthorne	0
Tree [5]	8	Todd	6
Tree [6]	0	Youngman	0
Tree [7]	0	Jones	0
Tree [8]	0	Ravage	0

- The pointer value 0 indicates a 'nil' pointer, thus a node will be pointing to nothing.
- Tree[1].Left = 2
- Tree[Tree[1].Left].Right = 3
- Tree[6].Data = "Youngman"

3. # Outline Primitive Data Types

There are 8 primitive data types: *byte, short, int, long, float, double, char, boolean*
• Primitive data are only single values; they have no special capabilities.
• Primitive types are the most basic data types available within the Java language and they serve as the building blocks of data manipulation.

Such types serve only one purpose — containing pure, simple values of a kind with their operations predefined and no new operation can be defined for such primitive types.

In the Java type system, there are three further categories of primitives:

- **Numeric primitives:** ***short***, ***int***, ***long***, ***float*** and ***double***. These hold numeric data only and are associated with operations of simple arithmetic (addition, subtraction, etc.) or of comparisons (is greater than, is equal to, etc.)
- **Textual primitives**: ***byte*** and ***char***. These hold characters (that can be Unicode alphabets or even numbers) and are associated with operations of textual manipulation (comparing two words, joining characters to make words, etc.).
- **Boolean and null primitives**: ***boolean*** *and* ***null***.(true/false and/or none)

Category	Types	Size (bits)	Minimum Value	Maximum Value	Precision	Example
Integer	byte	8	-128	127	From +127 to -128	`byte b = 65;`
	char	16	0	2^{16}-1	All Unicode characters	`char c = 'A'; char c = 65;`
	short	16	-2^{15}	2^{15}-1	From +32,767 to -32,768	`short s = 65;`
	int	32	-2^{31}	2^{31}-1	From +2,147,483,647 to -2,147,483,648	`int i = 65;`
	long	64	-2^{63}	2^{63}-1	From +9,223,372,036,854,775,807 to -9,223,372,036,854,775,808	`long l = 65L;`
Floating-point	float	32	2^{-149}	$(2-2^{-23})\cdot 2^{127}$	From 3.402,823,5 E+38 to 1.4 E-45	`float f = 65f;`
	double	64	2^{-1074}	$(2-2^{-52})\cdot 2^{1023}$	From 1.797,693,134,862,315,7 E+308 to 4.9 E-324	`double d = 65.55;`
Other	boolean	--	--	--	false, true	`boolean b = true;`
	void	--	--	--	--	--

4. Demonstrate Familiarity with Standard Algorithms

An Algorithm describes how a problem is solved in terms of the actions to be executed, and it specifies the order in which the actions should be executed.

- Can be illustrated using Descriptions, Flowcharts, Pseudocodes, Structure diagrams

a. Descriptions:

- general statements which are not governed by any programming language, that are followed in order to complete a specific task.

Enter temperature in °C
Store the value in box C
Calculate the equivalent temperature in °F
Store the value in box F
Print the value of box C
and F End the program.

b. Pseudocodes:

- These are English-like statements, closer to programming language that indicates steps followed in performing a specific task.

Algorithm for initialising a one-dimensional numeric array:

```
DIM TestScores(9) As Integer
DIM Index As Integer
FOR Index = 0 TO 9
        TestScores(Index) = 0
NEXT Index
```

Algorithm for initialising a two-dimensional string array:

```
DIM Students(4,3) As String
DIM RowIndex, ColumnIndex As Integer
FOR RowIndex = 0 TO 4
        FOR ColumnIndex = 0 TO 3
                Students(RowIndex,ColumnIndex) = “” NEXT
        ColumnIndex
NEXT RowIndex
```

Serial search on an array

The following pseudo-co de can be used to search an array to see if an item X exists:

```
01              DIM Index As Integer
02              DIM Flag As Boolean
03              Index = 0
04              Flag = False
05              Input X
06              REPEAT
07              IF TheArray(Index) = X THEN
08              Output Index
09              Flag = True  10              END IF
11      Index = Index + 1
12      UNTIL Flag = True OR Index > Maximum Size Of TheArray
13      IF Flag = False THEN
14      Show Message “Item not found”
15      END IF
```

Note that the variable Flag (line 04 and 09) is used to indicate when the item has been found and stop the loop repeating unnecessarily (line 12 ends the loop if Flag has been set to True).

Control Structures

- These includes: simple ***sequence, selection*** and ***iteration.***

i. sequence:

- instructions are executed in the order they appear in a program without jumping any one of them up to the end of the program.

e.g:

Enter first number, A
Enter second number, B
C = A + B
Print C
Stop

ii. Selection Structure:

- allows one to choose the route to follow in order to accomplish a specific task.
- is written using the **IF ....Then...Else** statement or the **CASE** statement.

IF...THEN ...ELSE statement:

- allows the user to choose one from at least two routes of solving a problem.

e.g:

Enter first Number, A
Enter second number, B
IF A>B THEN
Print A is bigger
ELSE
Print B is bigger
ENDIF

Nested If…then….Else

Start
Enter "First Number", A
Enter "Second Number", B
Enter "Third Number", C
If A>B Then
If B >C Then
Print "A is the largest Number"
Endif
Endif
End

CASE Statement: This is an alternative to the IF...THEN...ELSE statement and is shorter. For example:

Enter first Number, A
Enter second number, B
*Enter operand (+, -, * /)*
CASE operand of:
"+":
C = A + B
"-":
C = A-B
"":*
*C = A*B*
"/":
C = A/B
ENDCASE
Print C
END

Sorting Algorithms

c) Quick Sort

- A very fast method of sorting elements in an array
- Involves splitting an array into two sub lists, then quicksort each sub list by splitting them into two sub list and quicksort each….. recursively
- Quicksort involves recursively calling the process of ***partitioning*** (splitting) an array
- Partitioning is the process of selecting a ***pivot*** and rearranging elements in such a way that elements greater than the pivot are to the right of the pivot and those less than the pivot are to the left thus you have two sub lists.
- Repeat the process until all elements are sorted.

Let us go through this example

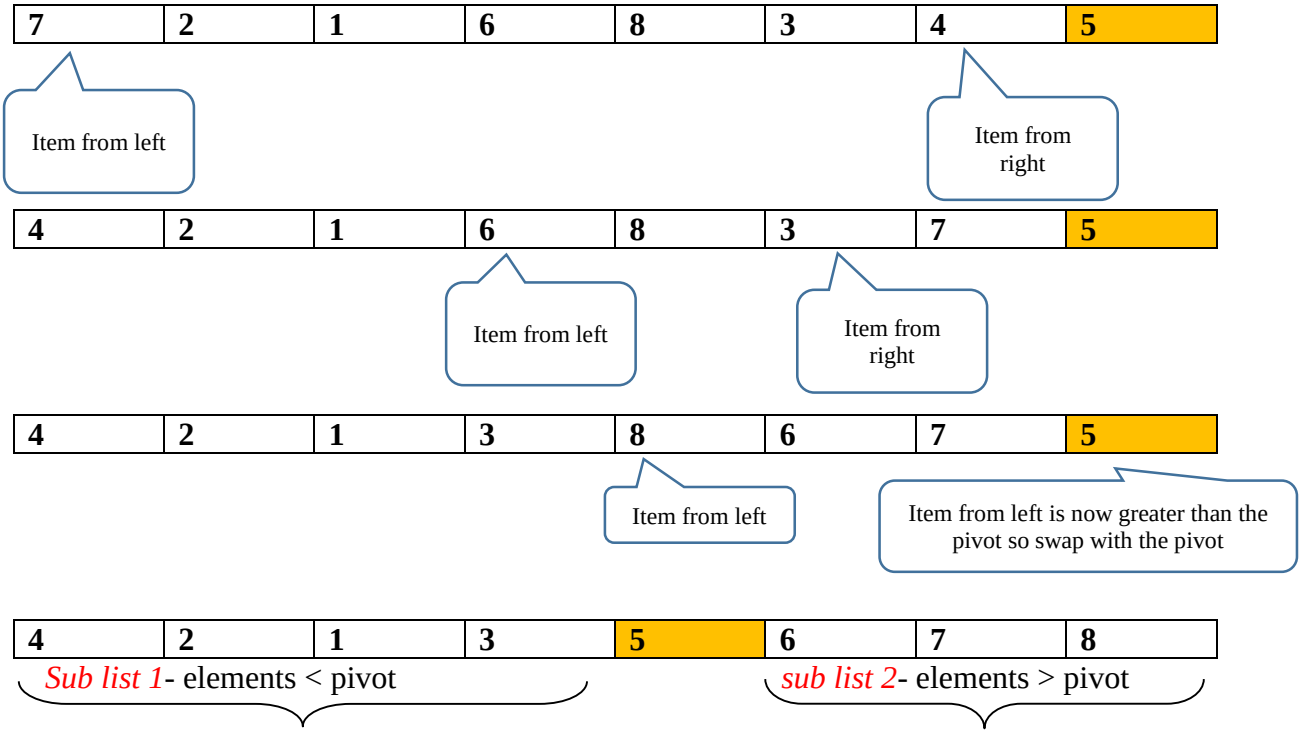
7	2	1	6	8	5	3	4

1

- First select a ***pivot*** (select any one of the elements and move it to the end of the array) say **5**

7	2	1	6	8	3	4	5

- Now to rearrange the elements we need to make swaps so that elements less than the pivot are to the left and those greater are to the right.
- Swap element from left which is greater than the pivot with element from the right which is smaller than the pivot, repeat the process until element from left is greater than the pivot, then swap the element with the pivot



maregat@mbizohighschool.ac.zw

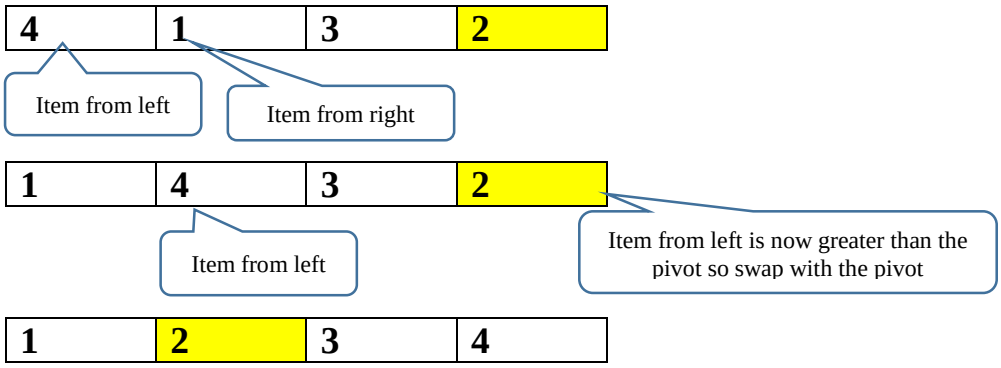
2 Now repeat the process for each sub list
Let's take sub list 1

4	2	1	3

- Select pivot, say 2 and move it to the end of the list

4	1	3	2

- Now to rearrange the elements we need to make swaps so that elements less than the pivot are to the left and those greater are to the right.
- Swap element from left which is greater than the pivot with element from the right which is smaller than the pivot, repeat the process until element from left is greater than the pivot, then swap the element with the pivot



- **Now we have a sorted sub list, do the same for the other sub list**

Then will have a sorted array

1	2	3	4	5	6	7	8

b. **Bubble sort**

- Sorting whereby the array is scanned from right to left multiple times (passes) until all elements are in their correct positions
- Compare each element with the adjacent, and if it is greater than the next element then swap the elements.

```
For k = 0 to N- 1      {counts the number of passes/ scans made to the array}
  For  i= 0 to N – 2 { use N-2 instead of N-1 so that at the end of the array you will be able to compare Array[i] and Array[i+1]}
       If Array[i] > Array[i+1] Then {compare adjacent elements}
       Swap (Array[i], Array[i+1])  { swapping element, i.e Array[i] = Array[i+1]}
       End if
  Next i
Next k
```

Example
Arrange the following numbers in ascending order using the bubble sort algorithm

17	18	2	11	0
[0]	[1]	[2]	[3]	[4]

1. – compare Array [0] with Array[1] {no swap}
   - Compare Array [1] with Array[2] {swap}
   - Compare Array [2] with Array [3] {swap}
   - Compare Array [3] with Array [4] {swap}

- Therefore after the **first pass** the array will be like this

17	2	11	0	18

2. – compare Array [0] with Array[1] {swap}
   - Compare Array [1] with Array[2] {swap}
   - Compare Array [2] with Array [3] {swap}
   - Compare Array [3] with Array [4] {no swap}

- After the 2nd pass

2	11	0	17	18

3.– compare Array [0] with Array[1] {no swap}
   - Compare Array [1] with Array[2] {swap}
   - Compare Array [2] with Array [3] {no swap}
   - Compare Array [3] with Array [4] {no swap}

- After the 3rd pass

2	0	11	17	18

4.– compare Array [0] with Array[1] {swap}
   - Compare Array [1] with Array[2] {no swap}
   - Compare Array [2] with Array [3] {no swap}
   - Compare Array [3] with Array [4] {no swap}

- After the 4th pass i.e k = **N-1**

0	2	11	17	18

c. Insertion Sort

- Items are copied from unsorted array to a new sorted array
- Each element is inserted into the right place so that the output array is always sorted
- Considerably faster than bubble sort but slower than quick sort

Searching Algorithms

a) Linear search

- searching data items one by one until item is found or end of list reached
- this involves comparing each item in the list with the item sought for match, if match then item found

Start at the beginning of list
Repeat
Test next item for a match
Until item found or end of list reached

b) Binary Search

- Search through an ordered array and is much faster than linear search.
- Divides the array into 3 parts: ***middle*** item, ***lower*** part and ***upper*** part
- The middle item is compared with the item sought for match, if they match them item found.
- If middle item is less than item sought (middle < itemsought), then the lower half of the array is discarded, it will be of no interest.
- Therefore, the number of items search is reduced by half, repeat the process until the las item is examined, with either the upper half or lower half of the items searched being discarded at each pass.

```
BinarySearch (Low, Top, ItemSought)
Low = 0
Top = n – 1                    {n = number of items in the array}
While low <=Upper
    Mid = (Low + Top)/2
    If A[Mid] = ItemSought Then    {compare middle with ItemSought, if equal them search found}
    Search found = True
    Else If ItemSought < A[Mid] Then
            Top = Mid – 1          {Discard upper half, if ItemSought is less than middle}
    Else
            Low = Mid + 1          {Discard lower half if ItemSought is greater than middle}
    Endif
End While
```

Example:

Given an array of integers **2,4,6,7,11,13,19,21,27,29**. Use Binary search to search for **19** in the array.

Soln:

2	4	6	7	11	13	19	21	27	29
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

Low = **0**
Top = 10 -1 = **9**
ItemSought = **19**

1 Mid = (0+9)/2 = 4,5 = **5** {round off to get the whole number part}

2	4	6	7	11	13	19	21	27	29
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

- Compare the item at index 5 with ItemSought (13 and 19)
- Mid (13) is less than ItemSought (19), therefore discard the lower half,
- low becomes **6** (mid +1) and Top remains **9**

19	21	27	29
[6]	[7]	[8]	[9]

2 Mid = (6+9)/2 = 7,5 = **8** {round off to get the whole number part}

19	21	27	29
[6]	[7]	[8]	[9]

- Compare the item at index 8 with ItemSought (27 and 19)
- Mid (27) is *greater* than ItemSought (19), therefore discard the upper half,
- low remains **6** and Top becomes **7** (mid -1)

19	21
[6]	[7]

3 Mid = (6+7)/2 = 6,5 = **7** {round off to get the whole number part}

19	21
[6]	[7]

- Compare the item at index 7 with ItemSought (21 and 19)
- Mid (21) is *greater* than ItemSought (19), therefore discard the upper half,
- low remains **6** and Top becomes **6** (mid -1)

19
[6]

4 Mid = (6+6)/2 = **6**

- Compare the item at index 6 with ItemSought (19 and 19), Mid = ItemSought therefore item found

Trace Table

Low	Top	Mid	Found
0	9	5	False
6	9	8	False
6	7	7	False
6	**6**	**6**	**True**

5. Use Example to Demonstrate Recursion

RECURSION

A recursive function or procedure occurs when the procedure calls itself other than calling another procedure.

Recursion can be used when finding factorial of a number. For example

```
Function Factorial (n)
    If n=1 then
        Return 1
    Else
        Return n * Factorial (n-1)
    End if
End Function
```

A recursive structure has two important features:

- It calls itself
- It must have a terminating condition (n=1 in the above example), otherwise it will not stop calling itself (runs forever). It uses the *if condition* (not a while) to specify the terminating condition.

The following is a recursive

```
procedure: Procedure Squares (Low, High)
If Low ≤ High Then
        Print (Low * Low)
        Squares (Low + 1, High)
End If
End Procedure
```

- If a recursive method is called with a base case, the method returns a result. If a method is called with a more complex problem, the method divides the problem into two or more conceptual pieces. Because this new problem looks like the original problem, the method launches a recursive call to work on the smaller problem.
- Both iteration and recursion are based on a control structure: Iteration uses a repetition structure; recursion uses a selection structure.
- Both iteration and recursion involve repetition: Iteration explicitly uses a repetition structure; recursion achieves repetition through repeated method calls.
- Iteration and recursion each involve a termination test: Iteration terminates when the loop continuation condition fails; recursion terminates when a base case is recognized.
- Iteration and recursion can occur infinitely: An infinite loop occurs with iteration if the loop continuation test never becomes false; infinite recursion occurs if the recursion step does not reduce the problem in a manner that converges on the base case.
- Recursion repeatedly invokes the mechanism, and consequently the overhead, of method calls. This can be expensive in both processor time and memory space.

Iteration / Repetition/looping:

A control structure that repeatedly executes part of a program or the whole program until a certain condition is satisfied.
Iteration is in the following forms: FOR...NEXT LOOP, REPEAT... UNTIL Loop and the WHILE...ENDWHILE Loop.

a. For...Next Loop: repeatedly executes the loop body for a specified number of times. The syntax of the For...Next loop is as follows:

```
FOR {variable} = {starting value} to {ending value} DO
        Statement 1
        Statement 2          loop body
        ..............
NEXT {variable}
```

- is appropriate when the number of repetitions is known well in advance, e.g. five times.

An example of a program that uses the **For...Next** loop is as follows:

```
Sum, Average = 0
FOR K = 1 to 5 DO
        Enter Number
        Sum = Sum + number
NEXT K
Average = Sum/5
Display Sum, Average
End
```

b. Repeat...Until Structure: repeatedly executes the loop body when the condition set is FALSE until it becomes TRUE. The number of repetitions may not be known in advance and the loop body is executed at least once. The syntax is as follows:

```
Repeat
        Statement 1
        Statement 2
        ..............
Until {Condition}
```

For example

```
Sum, Average, Count = 0
Repeat
        Enter Number (999 to end)
        Sum = Sum + Number
        Count = count +
1 Until Number = 999
Average = Sum / count
Print Sum, count, Average
End
```

C. While ... Do Structure

A looping structure in which the loop body is repeatedly executed when the condition set is TRUE until it becomes FALSE. It is used when the number of repetitions is not known in advance. The condition set is tested first before execution of the loop body. Therefore, the loop body may not be executed at all if the condition set is FALSE from start. The syntax of the WHILE…ENDWHILE structure is as follows:

WHILE {condition}
Statement 1
Statement 2 } loop body
.....
ENDWHILE

An example of the program is as follows:

Sum, Count, Average = 0
WHILE Count < 6 DO
Enter Number
Sum = Sum + number
Count = count + 1
ENDWHILE
Average = Sum/count
Display sum, count, average
END

The word **WEND** can be used to replace the word **ENDWHILE** in some structures and therefore is acceptable. The word **Do, after the condition** is optional.

Differences between the Repeat...Until and the While…ENDWHILE structures

	Repeat Until Loop	**While Endwhile Loop**
1	Loop body is executed when the condition set is FALSE until it becomes TRUE	Loop body is executed when the condition set is TRUE until it becomes FALSE
2	Loop body is executed at least once	Loop body may not be executed at all
3	Condition is tested well after execution of loop body	Condition is tested before execution of loop body

6. Design Algorithms for A Given Situation

An Algorithm describes how a problem is solved in terms of the actions to be executed, and it specifies the order in which the actions should be executed. To design an algorithm that serves the purpose, the following steps can be considered:

- *identify and/or declare variables* } **input**
- *assign values to variables/ input* }
- *manipulate variables* } **process**
- *display output* } **output**

NB:
By definition a computer is an electronic device capable of accepting **input**, **process** and produce **output**. Thus all programs designed to work with this gadget have to meet these requirements i.e

(1) accept input
(2) process the input
(3) display output

Example: A program is to be written which allows the user to enter numbers and it calculate the sum and average of the entered numbers. The user can enter as many numbers as he/she wishes; to end he/she has to enter "000" to end. Write an algorithm for the program.

```
Sum = 0
Average = 0          } declaring and initializing variable
Count = 0
Repeat
        Enter Number (000 to end)      } accept input

        Sum = Sum + Number
        Count = count + 1
Until Number = 000                     } manipulate variables
Average = Sum / count

Print Sum, count, Average     } display output
End
```

7. Analyze Algorithms for A Given Situation

Dry run a given algorithm

- This involves the programmer working through a program on paper, usually using a table called a 'trace table'.
- The programmer adds columns for any variables or expressions that are important and works through the program, line by line, filling into the trace table the values of variables and expressions as they change.
- By doing this, the programmer can spot the exact position when things start going wrong with the program - when variables suddenly contain unexpected values or expressions don't hold the expected state.

example

Here is a procedure used to produce times tables for children.

```
1.  Declare Num1, Num2, Multiplier, Answer, Counter As Integer
2.
3.  Multipler = 2
4.  Num1 = 1
5.  Num2 = 3
6.
7.  Do While Multiplier < 4
8.      For Counter = Num1 to Num2
9.          Answer = Counter * Multiplier
10.         Add_to_the_display: Counter & "Times" & Multiplier & " = " & Answer
11.     Next Counter
12      Multiplier = Mulitplier + 1
13.  Loop
```

Let us suppose that we want to analyse if this algorithm works as intended.

1. Create a **Tace Table** and
2. put all variables and expressions used by our algorithm in the first row
3. work through the program line by line

Line No.	Num1	Num2	Multiplier	Multiplier<4	Counter	Answer	Output
1							
2							
3			2				
4	1						
5		4					
6							
7				TRUE			
8					1		
9						2	
10							See Display
11							
8					2		
9						4	
10							See Display
11							
8					3		
9						6	
10							See Display
11							
12			3				
13							
7				TRUE			
8					1		
9						3	
10							See Display
11							
8					2		
9						6	
10							See Display
11							
8					3		
9						9	
10							See Display
11							
12			4				
13							
7				FALSE			
14							

Display screen

```
1 Times 2 = 2
2 Times 2 = 4
3 Times 2 = 6
1 Times 3 = 3
2 Times 3 = 6
3 Times 3 = 9
```

Let's work through the code ....

- We only put values in boxes when they change.
- Starting at line 1, no values are put into variables. Variables are declared on this line.
- There is no code on line 2.
- In line 3, the Multiplier is set to 2 so we write 2 in the box.
- In line 4, Num1 is assigned to 1 so we write 1 in the box.
- In line 5, Num2 is assigned to 3, so we write 3 in the Num2 box.
- There's no code on line 6.
- In line 7, the expression 'Multiplier < 4' Evaluates to TRUE. The Multiplier holds 2 and 2 is less than 4. So we write TRUE in the correct column and row.
- In line 8, we enter a FOR loop. Counter starts at 1, so we write 1 in the correct row and column.
- In line 9, we calculate answer to be Counter * Multiplier, or 2, so we put 2 in the correct box.
- In line 10, there is an output, so we write the output to our Display screen under the table. *1 Times 2 = 2* is written onto the display screen.
- When we get to Line 11, we jump back to line 8 because the FOR loop hasn't exceeded its upper limit yet.
- On line 8, Counter now becomes 2. We write 2 in the correct row and column.

Questions

i. a) The water level in a reservoir is controlled by a computer system. During normal operation the water level (W) is between the high water (H) and lower (L) marks. At these times the input valve (I) and output valve (O) are both open.
If the level reaches H then the input valve is shut off until the level falls below H again. If the level falls below L then the output valve is shut off until the level rises above L again. If the level falls below L for more than 1 hour the system sends an alarm signal to the operation. Using the variables W,H,L,I,O produce an algorithm to control the water level in the reservoir. [9]

b. The alarm signal together with all the other valves from the system is sent to a central control room. All the water supplies in the city are controlled from this central room by a single operator.

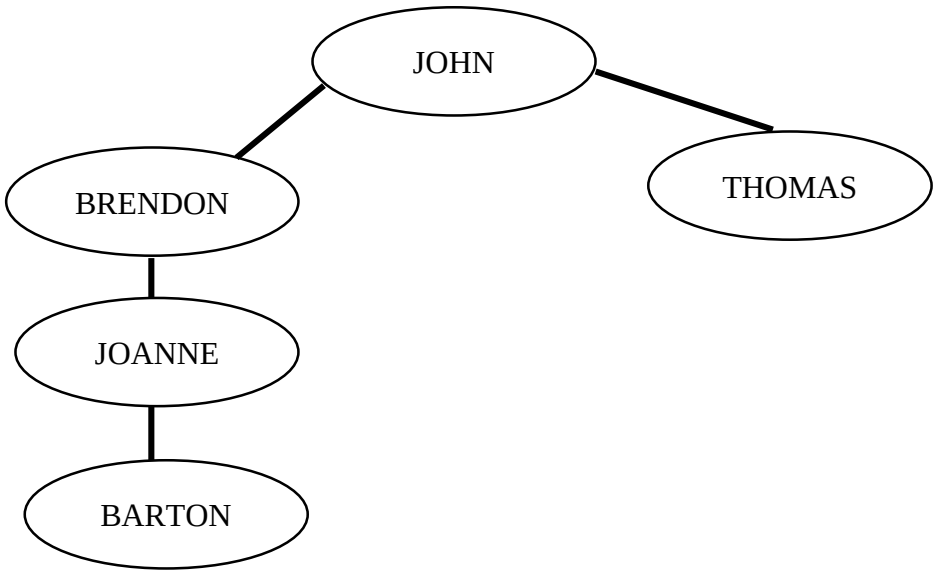
Explain the importance to the operator of good interface design, stating any features which should be considered. [5]

2 a) Describe algorithm for bubble-sorting integers into ascending order. [6]

(c) By making use of a suitable algorithm based on a bubble sort technique, show how the following data would be manipulated into its final ascending order:
7, 3 , 9 , 1 [6]

(c) explain why it is necessary to sort data on a computer file. [2]

4. A sort program inputs a unordered list of names and stores them in memory in the form of a binary tree. The figure below represents this data structure with the first six names entered.



(h) Define a binary tree.
(i) State
a The parent of BRENDON
b The name of the root of tree

(iii) Copy the diagram and add notes for SUSAN, WINNET, ANNA and JOSEPHINE on the list. [7]

3. (a) Distinguish between static and dynamic data structure. [2]
   (b) Outline any one benefit that would be gained from using a dynamic data structure to implement a binary tree. [1]
   (c) An initially empty tree has the following items added in the order: melon, peer, banana, orange.

   Draw the binary tree after the four items have been added. [3]

4. The following data items are to be entered into a binary search tree in the order given:

   Harare, London, Paris, Rome, Berlin, Amsterdam, Lisbon, Madrid, Lilongwe.

   (a) Draw a diagram to show to show how these values (data items) will be stored and circle the root node. [5]
   (b) If Madrid is being for in this binary tree, list the data items which will have to be accessed. [1]

5. Show how the following numbers can be sorted in ascending order using quick sort technique

   5; 2; 6; 1; 3; 4

   [10]

6. The details of a car part are stored in a binary tree according to the following algorithm.

```
Read Value New Part
        Start at Root Node
          While Node Not Empty Do
              If New-Part < Value At Node
                        Then follow Left Subtree
              Else
                        Follow Right Subtree
              Endif
         End While
              Insert New-Part At Node
        End
```

(j) Show the binary tree after the following values have been input
Rusape, Victoria Falls, Bulawayo, Triangle, Alaska, West Nichleson. [3]

(ii) Illustrate an algorithm using a flowchart for a program that accepts two numbers A and B. If A>B then display “A is bigger”, if A<B then display “B is bigger” else display “ A and B are equal” [6]

(iii) Arrange the following numbers in ascending order using the bubble sort algorithm

17 18 2 11 0 [4]

Enterprising

1. identify areas where computer science is applied

- Computers have great applications in the field of Information Technology, Physics, Medical sciences, Artificial intelligence, Robotics, Information Systems, Economics, Statistics, Financial Management, Nuclear Physics, Businesses, Human Resource Management etc.
- Computers are widely used by the businesses and organizations for effective management of information and resources in almost all departments.
- Marketing and advertising agencies rely on computer databases of customers and various other tools to track the needs of the customers.
- Human Resource Management Information Systems help organizations to keep record about the employees.
- Computers are used in networks which help individuals in the organizations to communicate with each other by sharing files and important documents.

2. evaluate the importance of e- Business

- E-business, also known as e-commerce, refers to conducting business via the internet.
- In the past, communications could take days if not weeks; now all business transactions can take place in only minutes.
- The benefits of e- businesses are vast and keep growing daily, these include:

- **Saves Money**

- small-business owners save a tremendous amount of money at startup, they save a lot of money when they don't have to pay for a storefront, utilities, building maintenance, and the time it takes to manage all that.
- products no longer have to go from manufacturer to warehouse to store to the customer's cart, they can go from the manufacturer to the customer's virtual cart, which saves on shipping costs and makes products cheaper overall.

- **Better Communication**

- E-business allows for conversations to happen quickly in several ways that facilitate understanding i.e phone or through a video chat, messaging and emailing.
- People who don't speak the same language can communicate through translation software programs.
- Any device that connects to the internet can be used as an invaluable e-business tool, i.e tablet, phone and computer
- Social media, email and teleconferencing mean that anywhere can be an office as long as there is internet connectivity.

- **Less Restrictive Hours**

- Its business 24/7, people can shop whenever they want and arrange for delivery or pickup.
- Information can be accessed at all hours and at everyone's leisure, which makes for a more pleasant experience overall for both the e-business and the customer.

- **Decision Making**

- enables management to make the right decision by conducting market research through online surveys, forums, blogs, group discussions using World Wide Web and of course through in-person interviews as well.
- These online tools not only provide real time responses from the potential audience but also ensure the accuracy of data by minimizing the risk of human errors.

- **Marketing and Business Growth**

- Digital Marketing enables promotion of products or services all over the world.
- It includes many concepts like search engine optimisation (SEO), pay per click (PPC), blogging, discussion forum, email shot, SMS, MMS, social media marketing and Smartphone app advertisement etc.
- These enables overall marketing, which covers public relation, advertising, promotion and sales which subsequently impact on business growth.

- **Customer Support and Satisfaction**

- Business success depends on knowing its customers' needs, trends, behaviours and satisfaction level.
- Internet Technology enables effective communication, which is the best tool to understand the customer demands, problems and their solutions.
- Enterprise organisations normally use customer relationship management systems (CRM) to hold valuable data for understanding customer behaviours and future needs.

- **Resource Management and Globalisation**

- cloud based ERP (Enterprise Resource Planning) solution enables managers to manage or monitor their organisational resources virtually anywhere in the world by using their personal computer, laptops, tablets or Smartphone.
- This concept has introduced the idea of globalisation. Most of multinational companies (Microsoft, Google, Amazon, McDonalds etc) in the world use these cloud based solutions to manage their virtual or physical offices and staff worldwide.

3. explain the elements of marketing

- **5Cs.**
  - **Company** - What the company is? Purpose? Vision?
  - **Customer** - Who is the target customer of the company?
  - **Collaborators** - Who are the allies of the company? Suppliers? People who can complement the selling of your goods or services?
  - **Competitors** - Who are your competitors? Evaluate them and their position on the market
  - **Context** - What is the business environment for the upcoming year?

- **The 4Ps of marketing are:**

  1. Product - Describe the product you are selling. What need are you meeting? What problem are you solving? Describe the services (offer sales support) that supplement your products.

2. Promotion - How do you intend to promote your product? Through newsletters? TV advertisements? Radio advertisements?

3. Price - What price do you sell your product at? What are the prices of similar products in the market?

4. Place - How do you intend to distribute the products. Through mass supermarkets? Concept stores? Small Mom-and-pop stores? Walmart?

4. design an ICT related business proposal
5. describe the role of Intellectual Property Rights

Intellectual property (IP) is a collective term used to describe new ideas, inventions, designs etc that are protected by copyright, patents, trademarks etc

Role of IP in Innovation

- IP system provides exclusive rights over the use of invention, design, brand, trademark etc
- As there are many players involved in facilitating the market success of an innovation, effective use of the tools of IP will play an important role in reducing risk for the players involved, who may then be able to reap acceptable returns for their participation in the process.
- IP plays an important role in facilitating the process of taking innovative technology to the market place.
- IP plays a major role in enhancing competitiveness of technology-based enterprises, whether such enterprises are commercializing new or improved products or providing service on the basis of a new or improved technology.

The different types of IP rights include trade secrets, utility models, patents, trademarks, geographical indications, industrial designs, layout designs of integrated circuits, copyright and related rights, and new varieties of plants.

- **Patent**
  - exclusive right granted for protection of an invention
  - Provides the owner with exclusive rights to prevent others from exploiting the invention.
  - An ***invention*** is considered as the generation of a new idea or knowledge, which aims to solve a specific technical problem.

- **Trademark**
  - A distinctive sign which distinguishes the goods or services produced by one enterprise from those of another
  - Any distinctive words, letters, numerical, drawings, colors, pictures, shapes, logotype, labels used to distinguish goods and services of different companies.
  - Trademark serves to:
    - Ensure that consumers can distinguish between products
    - Enable companies to differentiate between their products
    - Work as a marketing tool and basis for building brand image and reputation
    - Provide an incentive to companies to invest in maintaining or improving the quality of their products

- **Copyright**
- A set of exclusive rights granted by the law to the author or creator of an original work, including right to copy, distribute and adapt work.
- Copyright laws protect only the form of expression of ideas and not the ideas themselves.

6. analyse global trends in the field of computing

- **Cloud Computing**
- **Machine learning – Artificial Intelligence**

1. ***Spreading intelligence throughout the cloud***
   - Connected ***smart machines***, such as robots and autonomous vehicles, as a result of enhanced cloud architecture that can distribute and share machine intelligence
   - Developments in connectivity and cloud technologies are making it possible to distribute and share machine intelligence more easily, at a lower cost, and on a much wider scale than before.
   - connected to the cloud, smart machines will be able to use the powerful computational, storage and communication resources of state-of-the-art data centers.
   - ***Cloudification*** shifts the capabilities of these systems into a new sphere that includes complex problem-solving and decision-making on a mass-market scale.

2. ***Connect, store, compute... and share***
   - Shifting systems into the cloud enables communities of collaborating robots, machines, sensors and humans to process and share information.
   - Each new insight collected within a community can be shared instantly, which increases the effectiveness of collaborative tasks, and improves performance throughout the system, with a common awareness of system state shared by all participants, as well as a shared knowledge base.
   - distributed machine intelligence architecture offers lower implementation costs and sharing a backbone of unlimited computational power makes it possible to build lightweight, low-cost robots and smart machines that require a low level of control and a minimum amount of sensors and actuators.

3. ***Smart and mobile capabilities virtually everywhere***
   - Intelligent clouds will create new value chains in many industry segments, but some of the forerunners include mining, agriculture, forestry and health care.
   - New opportunities will open up for all organizations and people involved in the supply chain from the manufacturer to the customer.

- **Self-managing devices**

1. **Combining sensory data with AI techniques enables the data from massive numbers of sensors to be merged and processed to create a higher-level view of a system.**

- Connected smart devices will change our lives in many ways.
- Evolving software and communications technology are shifting toward the creation of autonomous and self-managing devices.

- The Internet of Things (IoT) means automation and intelligence in everything that is connected.
- The connectivity allows objects to be sensed and actuated remotely, creating a bridge between the physical and digital world.

2. **It's the combination that triggers the effect**
- Beyond the physical devices embedded with processors, software, sensors, actuators, and connectivity, it is the combination of sensory data and AI that enables more effective and accurate interactions.
- From a connectivity perspective, two distinct and different use cases emerge i.e the massive machine-type communication (massive MTC) that can support millions of connected devices such as energy meters and logistics tracking and
- the critical machine-type communication (critical MTC), which entails real-time control and automation of dynamic processes in various fields such as vehicle-to-vehicle, vehicle-to-infrastructure, high-speed motion, and process control. Critical parameters to enable the performance required are network latency below milliseconds, ultra-high "five nines" (99.999 percent) reliability.
- The future network architecture needs to cater for both MTC scenarios.

3. **Key technology advancements**
- On the device side, the key technology driver is the evolution of sensors, actuators, processors, memories, and batteries.
- Beyond conventional electronics, we will see implementations of nanoscale technologies based on thin-film, graphene, and quantum sensors.
- Another emerging key technology is that of an advanced software toolbox leveraging advanced analytics, machine learning, and knowledge management with processing capabilities of real-time streaming data.
- Intelligent control logic is another interesting area as there is an increasing need for standardized platforms and software protocols.

The future role of networks

- Internet of Things (IoT) devices enable us to monitor sensors and automate a lot of processes and the added intelligence needed is a feature that will mainly be embedded in the network itself.
- For IoT technology to live up to its promise and be applied on a massive scale throughout society, it must be built on a secure, global, telecom-grade network that is based on common standards which will also ensure a healthy competitive and innovative ecosystem.
- As for 5G, such an underlying network infrastructure is already in place – ready to show how well it is scaling and how its cost-efficiency properties support IoT applications.
- 5G offers both super-high bandwidth with ultra-low latency and extreme battery life for devices.
- By combining cloud intelligence with a powerful but energy-efficient wireless connection, even very simple and inexpensive devices can be made smart and generate great business value.

Questions

1. A software company has developed a new product for industrial usage.
   (a) Define *e-business*
   (b) Outline how any organization can use any 3 Ps of marketing in launching and marketing this newly developed product.

References

1. *Zimsec Computer Science Syllabus 2015 -2022*
2. *Zimsec Computer Science Specimen Papers 2017*
3. *A level Computers Module – Kapondeni T*
4. *A level Computing 5th Edition – PM Heathcote & S langfield*
5. *Internet Sources*