ZNOTES // A-LEVEL SERIES

visit www.znotes.org

Updated to 2017-2019 Syllabus

CIE A2-LEVEL COMPUTER SCIENCE 9608

SUMMARIZED NOTES ON THE THEORY SECTION

TABLE OF CONTENTS

3.1 Data representation

3.1.1 User-defined data types

- User-defined types make the program:
- Easier to understand
- Less error-prone
- Non-composite data type: A user-defined data type which does not involve reference to another type.
- Note: Non-composite data-types are required to be defined(explicitly) before being used.
- Enumeration: The complete, ordered listing of all the elements in a collection.

```
TYPE
<Identifier> = (<Element1>, <Element2>
.)

Eg:-
TYPE
Time = (Morning, Afternoon, Evening,
Night)
DECLARE Time1: Time
Time1 ← Afternoon
IF Time1>= Morning THEN
 Breakfast = TRUE
ENDIF
```

- ○ Pointer: A value that refers to (or "points to") a memory location elsewhere

```
TYPE
<Identifier> = ^<Data-Type>

Eg:-
TYPE
MyPointer = ^String
```

The declaration of the variable does not require the caret symbol (^).

```
DECLARE NameAddress: MyPointer
DECLARE Name: STRING
```

Accessing the value stored in the address pointed by the pointer

```
Name ← NameAddress^
```

- Composite data type: A user-defined data type which refers to one or more other data types.
  - ○ Set: A collection of particular values with no repetitions in any order.
  - ○ Record: A data type that combines different built-in data types in a record like structure.

```
TYPE
<Identifier>
DECLARE <Identifier> : <Data-Type>
DECLARE <Identifier> : <Data-Type>
ENDTYPE

Eg:-
TYPE
Student
DECLARE Name : STRING
DECLARE Level : INTEGER
DECLARE CS : BOOLEAN
ENDTYPE

DECLARE Student1 : Student
Student1.CS ← TRUE
Student1.Level ← A2
```

- **Object:** an instance of a class – a data type in which a record and methods that act on its properties are combined into one.

{S15-P31} **Question: 3**

(b) Annual rainfall data from a number of locations are to be processed in a program.

The following data are to be stored:

- location name
- height above sea level (to the nearest meter)
- total rainfall for each month of the year (centimeters to 1 decimal place)

A user-defined, composite data type is needed. The programmer chooses LocationRainfall as the name of this data type.

A variable of this type can be used to store all the data for one particular location.

(i) Write the definition for the data type LocationRainfall.

Solution:

Define the data type in structured English.

TYPE

LocationRainfall [1]

Now choose appropriate names for the variables and declare them.

DECLARE LocationName : STRING [1]

DECLARE HeightAboveSeaLevel : INTEGER [1]

Note: We choose Integer not the Float data type to store the height above the sea level since it was rounded to the nearest meter.

Since total rainfall for each month of the year needs to be stored under a single variable an array is required.

DECLARE TotalMonthlyRainfall : ARRAY[1..12] OF REAL [1+1]

Arriving at the final answer,

```
TYPE
LocationRainfall
DECLARE LocationName : STRING
DECLARE HeightAboveSeaLevel: INTEGER
DECLARE TotalMonthlyRainfall : ARRAY[1..12] OF
REAL
ENDTYPE
```

3.1.2 File organization and access

- Terminology:
  - **Field:** a single piece of data.
  - **Record:** A collection of fields containing data values.
- **File organization:** The way in which fields are organized/structured in a file.
  - **Serial file organization:** File organization in which the records are in no particular order.

    Eg: - Applications error-log file

Pros	Cons
Cheap	Requires a preset format for input and output of data
Fast	Slow access rate

  - **Sequential file organization:** File organization in which the records are ordered in some way by means of a key field - a field that consists of unique and sequential values.

    Eg: - Vinyl Albums

Pros	Cons
Suitable for files with long-term use	Adding/Deleting/Editing requires making a new file

  - Random file organization: File organization in which the data is organized using a record key can be accessed randomly.

    Eg: - Employee Database

Pros	Cons
Faster access speed	Less efficient in the use of storage space

- **File access:** A method by which a record/field in a file is read from or written to.
  - **Sequential access:** A method in which all records are accessed sequentially (one after another).

Eg: - Magnetic Tape

Pros	Cons
Simple organization	Difficult to update or delete individual fields/records, since it requires making a new file.
Easy to modify (can be modified using text editors)	Long access time since read and write operations happen sequentially.

- Direct access: A method in which records are accessed directly/randomly.

Eg: - CD

Pros	Cons
Easier to update or delete individual fields/records.	Easily damaged

	Sequential access	Direct access
Serial files	✓	✗
Sequential files	✓	✗
Sequential files with the index file	✗	✓
Random files	✗	✓

{S16-P32} **Question: 4**

(b) A bank has a very large number of customers. The bank stores data for each customer. This includes:

- unique customer number
- personal data (name, address, telephone number)
- transactions

The bank computer system makes use of three files:

- A– a file that stores customer personal data. This file is used at the end of each month for the production of the monthly statement.
- B– a file that stores encrypted personal identification numbers (PINs) for customer bank cards. This file is accessed when the customer attempts to withdraw cash at a cash machine (ATM).
- C – a file that stores all customer transaction records for the current month. Every time the customer makes a transaction, a new record is created. For each of the files A, B and C, state an appropriate method of organization. Justify your choice.

(i) File A organization & justification [3]
(ii) File B organization & justification [3]
(iii) File C organization & justification [3]

Solution:

(b)

(i) Sequential. Since all the customers get a statement (high-hit rate). Suitable for batch processing of the records since the records will be processed one after the other. File organized using customer's unique ID (as a primary key field)

(Or)

Serial [1]

Since all customers need to get the statement ... // high hit rate [1]

Suitable for batch processing of the records // the records will be processed one after the other [1]

Order not important [1]

(ii) Random [1]
Since the transaction requires real-time [1] processing
It requires fastest access to data [1]

(iii) Serial [1]
As each new record is appended, [1]
transactions are recorded in a chronological order. [1]

3.1.3 Real numbers and normalized floating-point representation

- **Real number:** A number that contains a fractional part.
- **Floating-point representation**: The approximate representation of a real number using binary digits.
- **Format: Number = ±Mantissa × BaseExponent**
  - **Mantissa:** The non-zero part of the number.
  - **Exponent:** The power to which the base is raised to in order to accurately represent the number.
- Base: The number of values the number systems allows a digit to take. 2 in the case of floating point representation.

In general:

- The Mantissa and the exponent are stored in twos-complement form.

-1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$

Alternatively

-2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}

8-bit Mantissa

-128	64	32	16	8	4	2	1

8-Bit Exponent

{SP15-P3} **Question: 1**

Floating-point is to be used to represent real numbers with:

- 8 bits for the mantissa, followed by
- 4 bits for the exponent
- two's complement used for both mantissa and exponent

(a)

(i) Consider this binary pattern.

0	1	1	0	1	0	0	0		0	1	0	0

What number is this in denary? Show your working. [3]

Solution:

Converting the Mantissa from binary to denary by adding the values for the mantissa

Mantissa = $2^{-1} + 2^{-2} + 2^{-4}$

$= \frac{1}{2} + \frac{1}{4} + \frac{1}{16}$

Since 2 & 4 are factors of 16, we take 16 as the denominator.

$\Rightarrow \frac{8}{16} + \frac{4}{16} + \frac{1}{16}$

$= \frac{+13}{16}$ [1]

Convert the Exponent

Exponent = +4 [1]

Now apply the values to the formula

Number = Mantissa × Base$^{\text{Exponent}}$

Number = $\frac{+13}{16} \times 2^4$

Number = +13 [1]

{W15-P32} **Question: 1**

(a)

(ii)Give the normalised binary pattern for +3.5. Show your working.

(iii)Give the normalised binary pattern for -3.5. Show your working.

Solution:

(ii) Take the number 3.5 and split it into whole and fractional parts

$3 = 3 + 0.5$

Now convert them into binary digits

$3_{10} = 11_2$

Without using the two's complement form

$0.5_{10} = .1_2$

Note: The point is used to specify that the following is a fractional part

Now combine the both,

$3.5_{10} = 11.1_2$ [1]

In order for this number to be in its normalised form, it must start with 0 followed by a 1 bit, since it being a positive number.

Moving the binary point two places backward

$11.1_2 = 0.111_2 \times 2^2$ [1]

Now converting the above expression to binary

$0.111_2 \times 2^2 = 01110000\ 0010_2$ [1]

- **Normalised form:** A particular format of the floating point representation which allows it to attain maximum precision.
- **Normalisation process**
  - Positive Number
  - Shift the bits in the mantissa leftward until the most significant bits are 0 followed by 1.
  - Negative Number
  - Shift the bits in the mantissa leftward until the most significant bits are 1 followed by 0.
- **Note:** For each left shift, the LSB (Least Significant Bit) of the exponent is removed.
- In general, the Mantissa
  - For positive numbers begins with a 0 followed by 1.
  - For positive numbers begins with a 0 followed by 1.
- Normalization is done to:
  - Save memory (efficiency)
  - Provide the most accurate representation of the number.

	Range of the values represented	Accuracy of the values represented
Increasing the number of bits allocated for Mantissa	↓	↑
Increasing the number of bits allocated for the Exponent	↑	↓

{W15-P31} **Question: 1**

(a)

(i)

Mantissa

0	0	1	0	1	0	0	0

Exponent

0	0	1	0	1	0	0	0

(ii) Explain why the floating-point number in **part (a) (i)** is not normalised. [2]

(iii) Normalize the floating-point number in **part (a) (i)**. [2]

Solution:

(a)

(ii) As a positive number's mantissa starts with zero [1]

And is followed by one [1]

(iii)Shifting all the bits of the Mantissa towards the left [1]

00101000 00000011 = 01010000 00000010

- **Underflow:** a scenario in which there is not enough bits to represent the number, since it being so small.

Eg: - Representing -64 for a 7-bit exponent

Exponent

1	0	0	0	0	0	0	0

- **Overflow:** a scenario in which there is not enough bits to represent the number since it being so big.

- **Eg: -** Representing +16 for a 4-bit exponent

0	0	0	0

0	1	0	0	0

- **Approximation:** Despite normalisation process, the number stored in a computer will always approximate the real number it represents. This is due the nature of the computer – being composed of binary values (1s and 0s).

{W15-P32} **Question: 1**

(c) A student enters the following expression into an interpreter:

```
OUTPUT (0.1 + 0.2)
```

The student is surprised to see the following output: 0.3000000000000001 Explain why this output has occurred.

Solution:

0.1 and 0.2 cannot be represented exactly in binary due to the rounding error present in the system by default. [1]

Therefore, it can be assumed that 0.1 and 0.2 are represented by a value just greater than their actual value. [1]

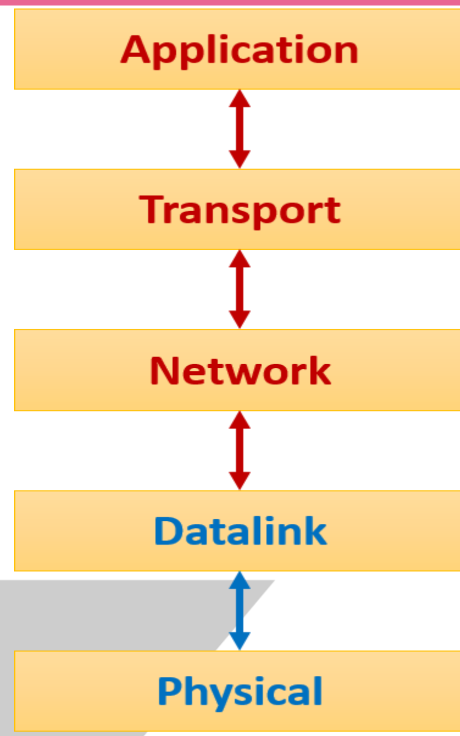
Thus, adding two representations together adds the two differences [1]

Summed difference significant enough to be seen.

3.2 Communication and Internet technologies

3.2.1 Protocols

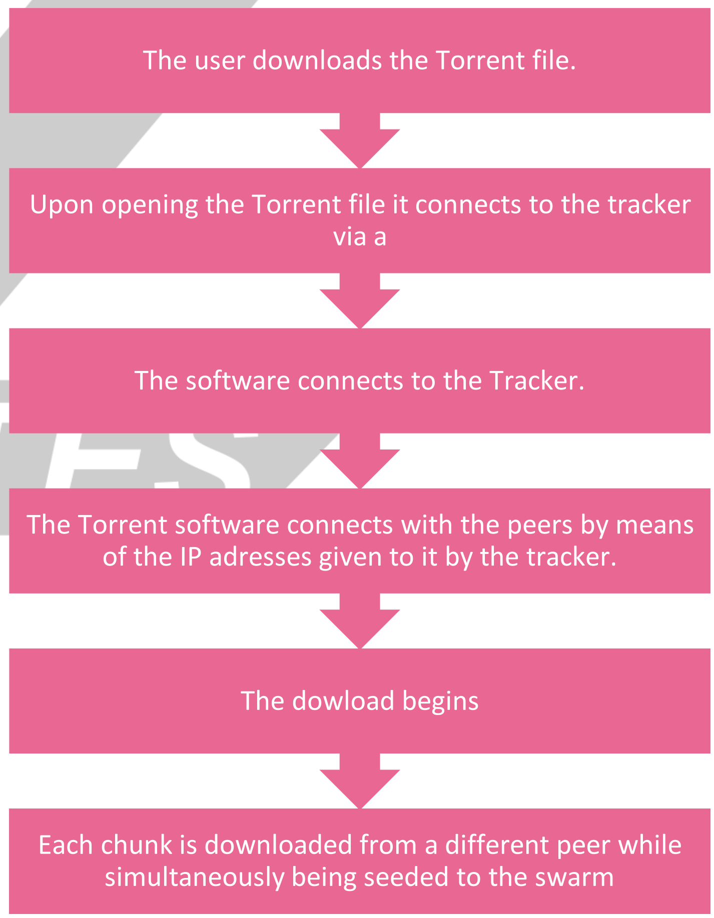
- **Protocol:** A set of rules governing communication between computers.
  - Ensures the computers that communicate understand each other.
- **MAC address:** A unique number assigned to each device's networking hardware across the world.
- **IP address:** A unique number assigned to each node/networking device in a network.
- **Port number:** A software-generated number that specifies an application or a process communication endpoint attached to an IP address.
- **IP:** Internet Protocol – Governs the sending and receiving of data over the internet.
- **TCP:** Transfer Control Protocol. A Protocol that controls the transfer of data segments.
- **TCP/IP Suite:** A common protocol used to send data over a network.
  - Protocols are split into separate layers, which are arranged as a stack.
  - They service each other thus maintaining the flow of the data.
- **Layer:** A division of the TCP/IP suite.
- **Stack:** A collection of elements/protocols/layers.



Layer	Purpose
Application	Encodes the data being sent
Network/Internet	Adds IP addresses stating where the data is from and where it is going
Link	Adds MAC address information to specify which hardware device the message came from and which hardware device the message is going to
Physical	Enables the successful transmission of data between devices

- When a message is sent from one host to another:
  - **Sender side: Application Layer**
    - Encodes the data in an appropriate format.
  - **Sender side: Transport Layer**
    - The data to be sent is broken down into smaller chunks known as packets
  - **Sender side: Network Layer**
    - IP addresses (sender and receiver) and a checksum are added to the header
  - **Sender side: Link Layer**
    - Formats the packets into a frame. These protocols attach a third header and a footer to "frame" the packet. The frame header includes a field that checks for errors as the frame travels over the network media.
  - **Sender side: Physical Layer**
    - Receives the frames and converts the IP addresses into the hardware addresses appropriate to the network media. The physical network layer then sends the frame out over the network media.
  - **Server/ Service Provider**
    - Re-Routes the packets according to the IP address
  - **Receiver side: Physical Layer**
    - Receives the packet in its frame form. It computes the checksum of the packet, and then sends the frame to the data link layer.
  - **Receiver side: Link Layer**
    - Verifies that the checksum for the frame is correct and strips off the frame header and checksum. Finally, the data link protocol sends the frame to the Internet layer.
  - **Receiver side: Network Layer**
    - Reads information in the header to identify the transmission and determine if it is a fragment. If the transmission was fragmented, IP reassembles the fragments into the original datagram. It then strips off the IP header and passes it on to transport layer protocols.
  - **Receiver side: Transport Layer**
    - Reads the header to determine which application layer protocol must receive the data. Then TCP o strips off its related header and sends the message or stream up to the receiving application.
  - **Receiver side: Application Layer**
    - Receives the message and performs the operation requested by the sender

- **Bit Torrent protocol:** A protocol that allows fast sharing of files via peer-to-peer networks.
  - Torrent file: A file that contains details regarding the tracker
  - Tracker: A server that keeps track of the peers
  - Peers: A user who is at the time downloading the same file as the
  - Swarm: A network of peers that are sharing the torrent – simultaneously downloading and uploading the file.
  - Seeding: The act of uploading a part of the file or the file itself as a whole after/while downloading
  - Leeching: The act of simply downloading a part of the file or the file itself on a whole and not seeding it during or after the download.
  - Seeders: Users who are currently seeding the file.
  - Leechers/Free-raiders: Peers who are currently leeching the file.



Other protocols:

Acronym	Protocol	Purpose
HTTP	Hyper Text Transfer Protocol	Handles transmission of data to and from a website
FTP	File Transfer Protocol	Handles transmission of files across a network
POP3	Post Office Protocol 3	Handles the receiving of emails
SMTP	Simple Mail Transfer Protocol	Handles the sending of emails

3.2.2 Circuit switching, packet switching and routers

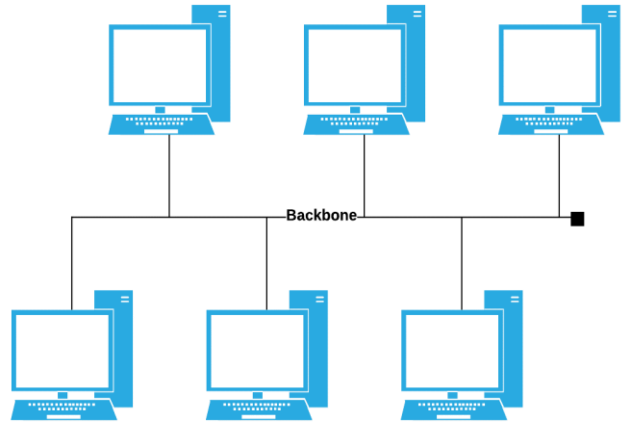
- **Circuit switching:** A method of data transfer in which the message is sent over a dedicated communication channel.

  Eg: - Landline Phone
- **Packet switching:** A method of data transfer in which the intended message is broken down into parts and is sent over whichever route is optimum in order to reach its destination.
  - Each packet travels through several other networks – "switching" between them in order to reach its destination.

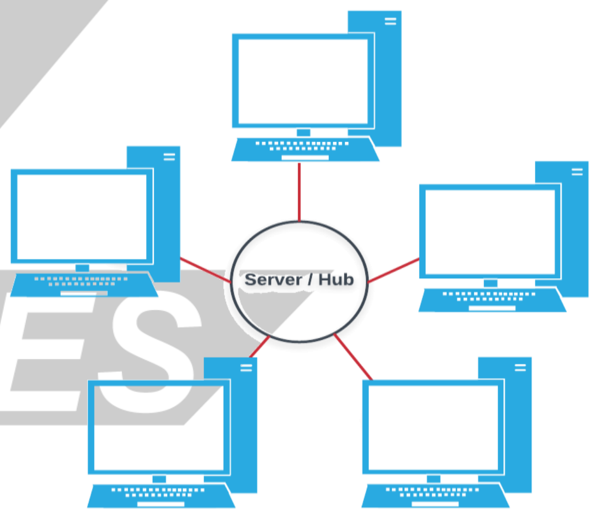
  Eg: - Internet
- **Router:** A device that connects two or more computer networks.
  - Directs the incoming packets to their receiver according to the data traffic in the network.

3.2.3 Local Area Networks (LAN)

- **Network topology:** A specific arrangement of networking devices to form a network.



- **Bus topology:** A network topology in which each workstation is connected to a main cable (backbone) through which the network is established.
  - The Backbone acts as the common medium, any signals sent or received go through the backbone in order to reach the recipient.



- **Star topology:** A network topology in which each workstation is connected to a central node/connection point through which the network is established.
  - The central node (hub) re-directs and directs the packets according to the data traffic and their recipient.
- **Wireless networks:** A computer network that uses wireless data connections between its network components.
- **Bluetooth:** A type of short-range wireless communication that uses

- **Wi-Fi OR IEEE 802.11x.**– A type of wireless communication that allows the users to communicate within a particular area/ access internet.

Component	Purpose in a LAN
Switch	Allows different networks to connect
Router	Directs the incoming packets into
Servers	Provides a medium for the storage, sharing of usage of files and applications for its users
Network Interface Cards (NICs)	Consists of the electronic circuitry required to communicate with other networks/devices.

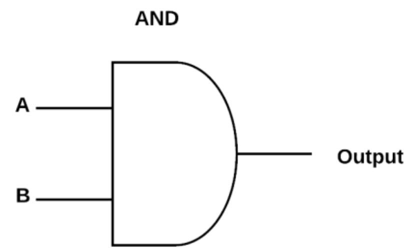
- **Ethernet:** an array of networking technologies and systems used in local area networks (LAN), where computers are connected within a primary physical space.
- **Collision detection and avoidance:**
- **CSMA/CD:** Carrier-Sense Multiple Access with Collision Detection is a collision detection and avoidance method predominantly used in early Ethernet technology.
  - It monitors the voltage levels of a cable/medium to detect a colilision.
- How CSMA/CD works:
  - Listens to the medium to check if it's busy
  - If not, then the host starts to transmit the signal
  - Once the transmission begins the current level is continuously monitored to check for collisions
  - If a collision is detected, then a special Jamming signal is sent to inform all the receiver that there was a collision

3.3 Hardware

3.3.1 Logic gates & circuit design

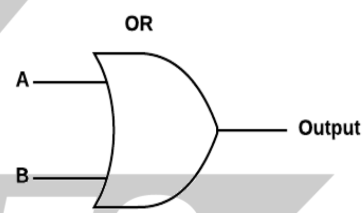
Logic gates: A component of a logical circuit that can perform a Boolean operation (logical function).

- **AND Gate:** $A.B = X$



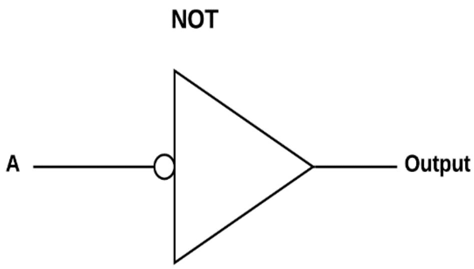
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

- **OR Gate:** $A + B = X$



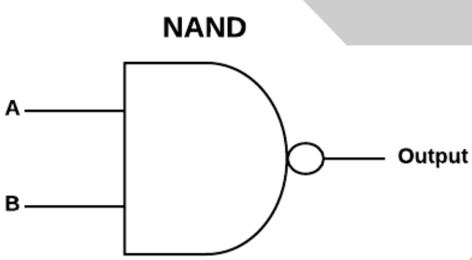
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

- **NOT Gate:** $\bar{A} = X$



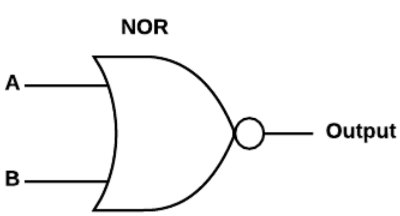
A	Output
0	1
1	0

- **NAND Gate:** $\overline{A.B} = X$



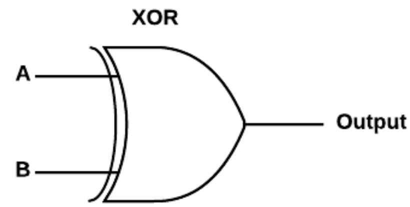
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

- **NOR Gate:** $\overline{A+B} = X$



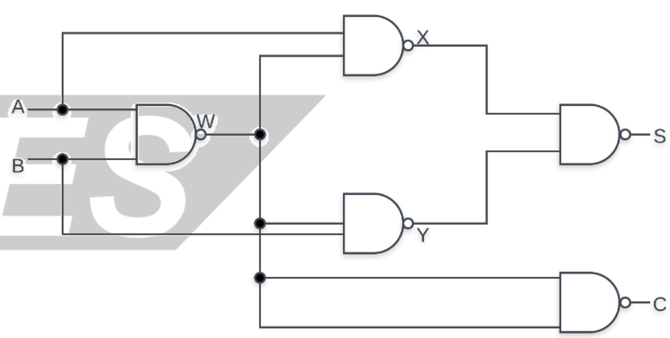
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

- **XOR Gate:** $A.\bar{B} + \bar{A}.B = X$



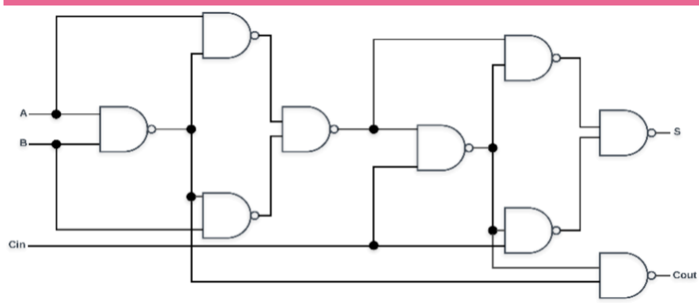
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

- **Logic circuits:** A circuit that performs logical operations on symbols.
- **Sequential circuit:** a circuit whose output depends on the input values and the previous output values. Eg: - Flip-flops (Section 3.3.4)
- **Combinational circuit:** a circuit whose output is dependent only on the input values
  - **Half-Adder:** A logic circuit that adds two bits together and outputs their sum.



Input		Output	
A	**B**	**S**	**C**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

  - **Full-Adder:** A logic circuit that adds multiple bits together and outputs their sum.



Input			Output	
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Legend:

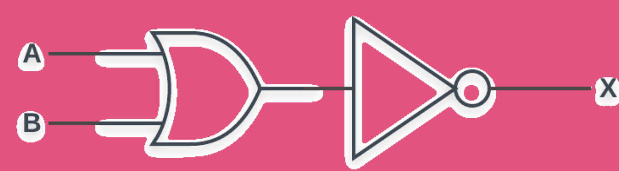
S	**Sum bit:** Stores the sum value of the two bits
C	**Carry bit:** Stores the carry over value of the two bits.

3.3.2 Boolean algebra

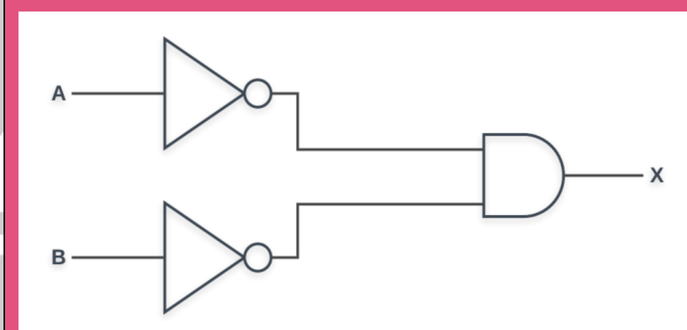
- **Double Complement:** $\bar{\bar{A}} = A$
- **Identity Law**
  - $1.A = A$
  - $0 + A = A$
- **Null Law**
  - $0.A = 0$
  - $1 + A = 1$
- **Idempotent Law**
  - $A.A = A$
  - $A + A = A$
- **Inverse Law**
  - $A.\bar{A} = 0$
  - $A + \bar{A} = 1$
- **Commutative Law**
  - $A.B = B.A$
  - $A + B = B + A$
- **Associative**
  - $(A.B).C = A.(B.C)$
  - $(A + B) + C = A + (B + C)$
- **Distributive Law**
  - $A + B.C = (A + B).(A + C)$
  - $A.(B + C) = A.B + A.C$
- **Adsorption**
  - $A.(A + B) = A$
  - $A + A.B = A$
- **De Morgan's Law**
  - $(\overline{A.B}) = \bar{A} + \bar{B}$
  - $(\overline{A + B}) = \bar{A}.\bar{B}$

{S15-P33}
Question: 5

a.
 i. Complete the truth table for this logic circuit:

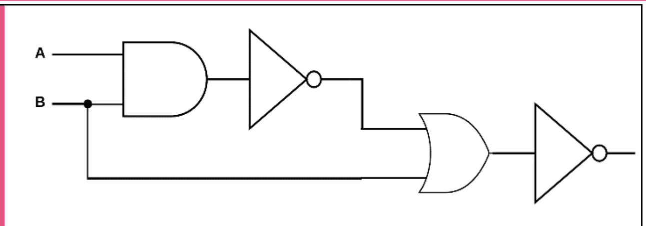


 ii. Complete the truth table for this logic circuit:



b. A student decides to write an equation for **X** to represent the full behaviour of each logic circuit.
 i. Write the Boolean expression that will complete the required equation for **X** for each circuit:
  - Circuit 1: **X** =
  - Circuit 2: **X** =
 ii. Write the De Morgan's Law which is shown by your answers to **part (a)** and **part (b)(i).**

c. Write the Boolean algebraic expression corresponding to the following logic circuit:



d. Using De Morgan's laws and Boolean algebra, simplify your answer to **part (c)**.

- Show all your working.

Solution:

- **Part a:**

  i. Assume that the letter "C" equates to the first function/logic gate.

- $A + B = C$

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

- $C = A + B$

$\bar{C} = X$

C	X
0	1
1	0
1	0
1	0

[1]

ii. Allocate unique letters for the output of each gate.

- $\bar{A} = C$

$\bar{B} = D$

- Now apply the Boolean operations on each input.

A	C
0	1
0	1
1	0
1	0

B	D
0	1
1	0
0	1
1	0

- Now apply the next gate's operation to the results.
- $D.C = X$

D	C	X
1	1	1
0	1	0
1	0	0
0	0	0

[1]

Part b:

i. Circuit 1: $\boldsymbol{X} = \overline{A + B}$ [1]

- Circuit 2: $\boldsymbol{X} = \bar{A}.\bar{B}$ [1]

ii. $\overline{A + B} = \bar{A}.\bar{B}$ [1]

Part c:

Group separate gates/results in brackets just like a normal mathematical equation.

$$(A.B)$$
$$\overline{(A.B)}$$
$$\overline{(A.B)} + B$$
$$\overline{\overline{(A.B)} + B} = X$$

Part d:

Using De Morgan's Law [1]

$$\overline{\overline{(A.B)} + B} = \overline{\overline{(A.B)}} + \bar{B}$$

Using Law Double Complement Law [1]

$$\overline{\overline{(A.B)}} + \bar{B} = (A.B)\bar{B}$$

Using the Associative Law:

$$(A.B).\bar{B} = A.(B.\bar{B})$$

Using the Inverse Law

$$A.(B.\bar{B}) = A.0$$

Using the Null Law [1]

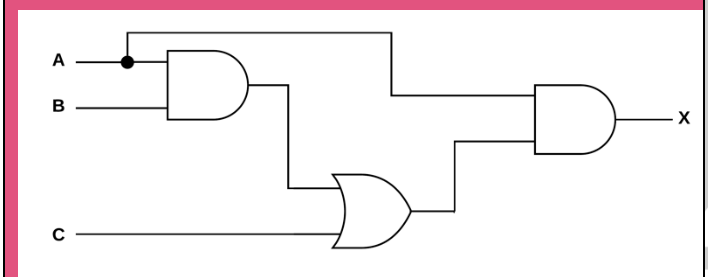
$$A.0 = 0$$

3.3.3 Karnaugh Maps

- **Karnaugh maps:** a method of obtaining a Boolean algebra expression from a truth table involving the
- Benefits of using Karnaugh Maps:
  - Minimises the number of Boolean expressions.
  - Minimises the number of Logic Gates used, thus providing a more efficient circuit.
- Methodology
  - Try to look for trends in the output, thus predict the presence of a term in the final expression
  - Draw out a Karnaugh Map by filling in the truth table values into the table
  - Column labelling follows Gray coding sequence
  - Select groups of '1' bits in even quantities (2, 4, 6 etc), if not possible then consider a single input as a group
  - Note: Karnaugh Maps wrap around columns
  - Within each group only the values that remain constant are retained

{S17-P31} **Question: 3**

Consider the following logic circuit, which contains a redundant logic gate.



(a) Write the Boolean algebraic expression corresponding to this logic circuit. [3]

(b) Complete the truth table for this logic circuit.

A	B	C	Working Space	X
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

[2]

(c)

(i) Complete the Karnaugh Map (K-map) for the truth table in **part (b)**.

		AB			
		00	01	11	10
C	0				
	1				

The K-map can be used to simplify the expression in **part (a)**. [1]

(ii) Draw loop(s) around appropriate groups to produce an optimal sum-of-products. [2]

(iii) Write a simplified sum-of-products expression, using your answer to **part (ii).** [2]

(a) $B.C$ [1]

$\bar{B} + B.C$ [1]

$X = A.\left(\bar{B} + (B.C)\right)$ [1]

(b) Assuming,

$B.C = Q$

B	C	Q
0	0	0
0	1	0
1	0	0
1	1	1
0	0	0
0	1	0
1	0	0
1	1	1

B	$\bar{B}$
0	1
0	1
1	0
1	0
0	1
0	1
1	0
1	0

$$\bar{B} + Q = P$$

$\bar{B}$	Q	P
1	0	1
1	0	1
0	0	0
0	1	1
1	0	1
1	0	1
0	0	0
0	1	1

$$A.P = X$$

A	Q	X
0	1	0
0	1	0
0	0	0
0	1	0
1	1	1
1	1	0
1	0	1
1	1	1

A	**B**	**C**	**X**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

1 mark for every correct 4 entries

(c)

(i) Fill in the inputs which result in an output of 1 with 1 in the table.

AB

		00	**01**	**11**	**10**
C	**0**	0	0	0	1
	1	0	0	1	1

[1]

(ii) Circle the "1" bits as pairs of twos, since it being the smallest unit to group continuously - both vertically and horizontally.

AB

		00	**01**	**11**	**10**
C	**0**	0	0	0	1
	1	0	0	1	1

1 mark for each circle

(iii) Now, consider the vertical oval. Keep only the terms that do no change, therefore C will be eliminated as it changes from $\bar{C}$ to C

$A.\bar{B}$ [1]

Consider the horizontal oval, apply the same technique

$A.C$ [1]

Now add the two terms to obtain an equation for x

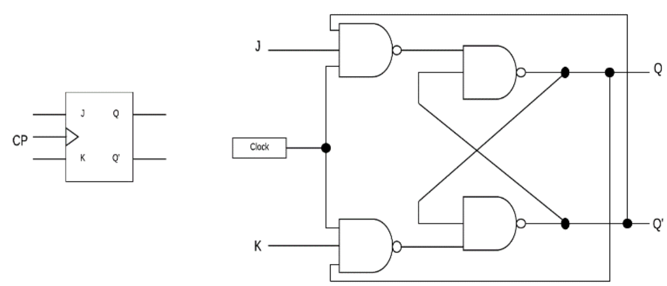
$$X = A.\bar{B} + A.C$$

3.3.4 Flip-flops

- SR flip-flop: SR(Set-Reset) flip-flop or "Latch"
  - Used as a storage device for 1 bit in the RAM, since it's values can be altered
  - **Issue:** When the both the input signals are 1 (invalid state) the flip-flop sets the value of Q and Q' to 0.

Input signals		Initial state		Final state	
S	**R**	**Q**	**Q'**	**Q**	**Q'**
0	0	1	0	1	0
1	0	1	0	1	0
0	1	1	0	0	1
0	0	0	1	0	1
1	0	0	1	1	0
0	1	0	1	0	1

- JK flip-flop: The J acts as a set input and the K as a clear input.
  - Fixes the issue with the SR flip-flop by means of a clock input to synchronise the inputs
  - When both the input signals are one, Q toggles.



J	K	Clock	Q
0	0	↑	Q unchanged
1	0	↑	1
0	1	↑	0
1	1	↑	Q toggles

- **Flip-flops** are used to build:
  - Data storage elements
  - Digital circuits

3.3.5 RISC processors

- **RISC:** Reduced Instruction Set Computers.
- **CISC:** Complex Instruction Set Computers.

RISC	CISC
Fewer instructions	More instructions
Simpler instructions	Complicated instructions
Small number of instruction formats	Many instruction formats
Single-cycle instructions whenever possible	Multi-cycle instructions
Fixed-length instructions	Variable-length instructions
Only load and store instructions to address memory	May types of instructions to address memory
Fewer addressing modes	More addressing modes
Multiple register sets	Fewer registers
Hard-wired control unit	Microprogrammed control unit
Pipelining easier	Pipelining much difficult

- **Pipelining:** Instruction level parallelism
  - Used extensively in RISC processor based systems to reduce the time taken to run processes
  - Multiple registers are employed
- **Interrupt handling in CISC and RISC Processors:**
  - As soon the interrupt is detected the current processes are paused and moved into registers
  - The ISR (Interrupt Service Routine) is loaded on to the pipeline and is executed.
  - When the interrupt has been serviced, the paused processes are resumed by bringing them back from the registers to the pipeline

3.3.6 Parallel processing

- **SISD**
  - Single Instruction Single Data stream
  - Found in the early computers
  - Contains single processor thus no pipelining
- **SIMD**
  - Single Data Multiple Instruction stream.
  - Found in array processors
  - Contains multiple processors, which have their own memory.

- **MISD**
  - Multiple Instruction Single Data stream
  - Used to sort large quantities of data.
  - Contains multiple processors which process the same data
- **MIMD**
  - Multiple Instruction Multiple Data.
  - Found in modern personal computers.
  - Each processor executes a different individual instruction.
- **Massively parallel computers**
  - Computers that contain vast amounts of processing power.
  - Has a bus structure to support multiple processors and a network infrastructure to support multiple 'Host' computers.
  - Commonly used to solve highly complex mathematical problems.

3.4 System software

3.4.1 Purposes of an operating system (OS)

- Optimizes use of computer resources
  - Implements process scheduling to ensure efficient CPU use
  - Manages main memory usage
  - Optimizes I/O
    - Dictates whether I/O passes through CPU or not
- Hides the complexities of the hardware
  - UI allows users to interact with application programs
  - Automatically provides drivers for new devices
  - Provides file system
    - Organizes physical storage of files on disk
  - Provides programming environment, removing the need for knowledge of processor functions
  - Provides system calls/APIs
    - Portability
- **Multitasking:**
  - More than one program can be stored in memory, but only one can have CPU access at any given time
  - Rest of the programs remain ready
- **Process:**
  - A program being executed which has an associated Process Control Block (PCB) in memory
    - **PCB:** a complex data structure containing all data relevant to the execution of a process
  - Process states
    - **Ready:** New process arrived at the memory and the PCB is created
    - **Running:** Has CPU access
    - **Blocked:** Cannot progress until some event has occurred
- Scheduling ensure that the computer system is able to serve all requests and obtain a certain quality of service.
- **Interrupt:**
  - Causes OS kernel to invoke ISR
    - The kernel may have to decide on priority
    - Register values stored in PCB
  - Reasons
    - Errors
    - Waiting for I/O
    - Scheduler halts process
- **Low-level scheduling:** Allocation specific processor components to complete specific tasks.
- **Low-level scheduling algorithms**
- **Preemptive:** Will stop the process that would have otherwise have continued to execute normally.
- **First-come-first-served**
  - Non-preemptive
  - FIFO(First In First Out) queue
- **Round-robin**
  - Allocates time slice to each process
  - Preemptive
  - Can be FIFO queue
  - Does not prioritize
- **Priority-based**
  - Most complex
    - Priorities re-evaluated on queue change
    - Priority calc. Requires computation
  - Criteria for priority time
    - Estimated time of execution
    - Estimated remaining time of execution
    - Is the CPU/IO bound?
    - Length of time spent in waiting queue

- **Paging:**
  - Process split into pages, memory split into frames
  - All pages loaded into memory at once
- **Virtual memory:**
  - No need for all pages to be in memory
  - CPU address space thus larger than physical space
    - Addresses resolved by memory management unit
  - Benefits
    - Not all of the program has to be in memory at once
    - Large programs can be run with or without large physical memory
  - Process
    - All pages on disk initially
    - One/more loaded into memory when process 'ready'
    - Pages replaced from disk when needed
      - Can be done with FIFO queue or usage-statistics based algorithm
- **Disk thrashing:** Perpetual loading/unloading of pages due to a page from disk immediately requiring the page it replaced.

3.4.2 Virtual machine

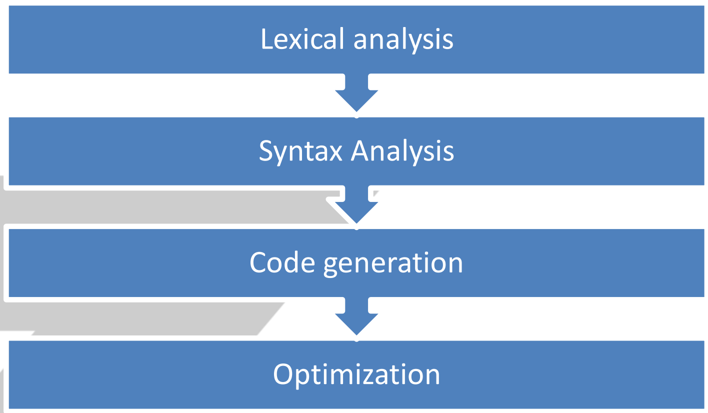
- **Virtual machine:**
  - Process interacts with software interface provided by the OS. This provides exact copy of hardware.
  - OS kernel handles interaction with actual host hardware

Pros	Cons
Allows more than one OS to run on a system	Performance drop from native OS
Allows multiple copies of the same OS	Time and effort needed for implementation is high

  - Used by companies wishing to use the legacy software on newer hardware and server consolidation companies

3.4.3 Translation software

- **Lexical analysis:** The process of converting a sequence of characters to a sequence of tokens.
  - **Tokens:** Strings with an assigned meaning
- **Syntax analysis:** The process of double-checking the code for grammar mistakes (syntax errors).
- **Code generation:** The process by which an intermediate code is generated after syntax analysis.
- **Optimization:** A process in which the code is edited to make improvements in efficiency.



- **For interpreters:**
  - Analysis and code generation run for each code line as above
  - Each line executed as soon as intermediate code generated
- **BNF:** Backus Naur Form
  - Can be used as a basis for an algorithm
  - Can be defined recursively

Notation	Meaning
::=	Defined by
|	OR
<x>	Meta variable

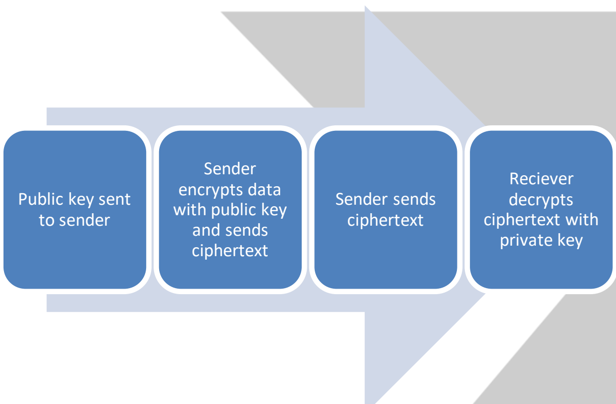
- **RPN:** Reverse Polish Notation
  - Used for representing algebraic expressions
  - Operator placed after operands
  - Example:

X + Y -> X Y +

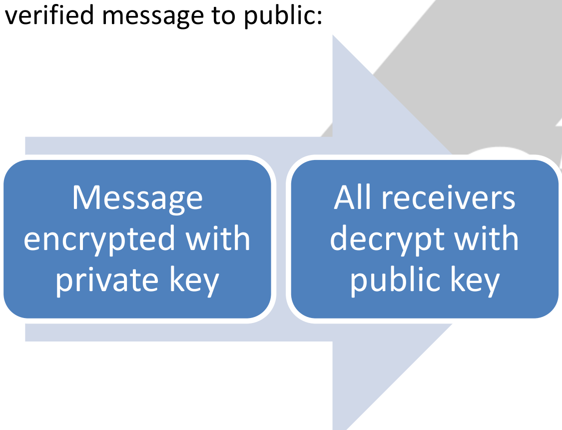
3.5 Security

3.5.1 Asymmetric keys and encryption methods

- **Plain text:** data before encryption.
- **Cipher text:** the result of applying an encryption algorithm to data.
- **Encryption:** the making of cipher text from plain text.
- **Public key:** encryption key which is not secret.
- **Private key:** encryption key which is a secret.
- **Asymmetric encryption:** Encryption where both a private and public key is used
  - Matched pair of keys
- Sending a private message:

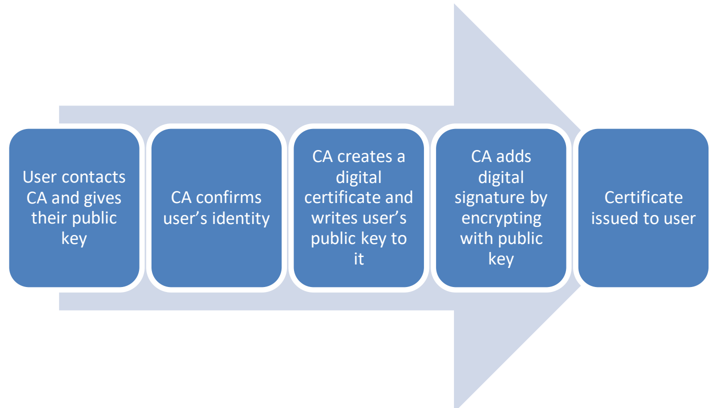


- Sending verified message to public:

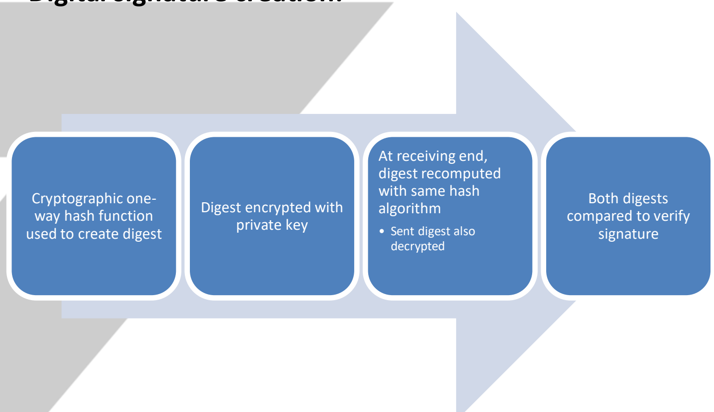


3.5.2 Digital signatures and digital certificates

- **Certificate acquisition:**



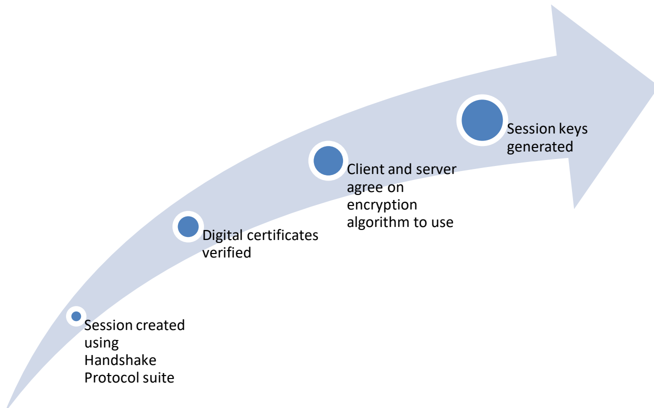
- **Digital signature creation:**



3.5.3 Encryption protocols

- SSL and TLS encryption protocols are used in client-server applications.
- Provides:
  - Encryption
  - Compression of data
  - Integrity checking

- **Connection process:**



- Used in online shopping and banking websites.

3.5.4 Malware

- **Virus:** tries to replicate inside other executable programs.
- **Worm:** runs independently and propagates to other network hosts.
- **Spyware:** collects info & transmits to another system.
- **Phishing:** email from seemingly legit source requesting confidential info.
- **Pharming:** setting up a bogus website that appears to be legit.

Malware	Vulnerabilities exploited
Virus	Executable files used to run or install software.
Worm	Shared networks
Spyware	Background processes
Phishing	Users mindset on considering emails from random addresses to be trustworthy
Pharming	Users mindset of relying on the websites user interface than URL for its validity.

Malware	Methods of restriction
Virus	ıstall and use an Anti-Virus software that runs daily scans.
Worm	etup a firewall to protect yourself from external networks.
Spyware	nstall and use real time ınti-Spyware protection.
Phishing	lways double check the website name.
Pharming	lways check the senders email address.

3.6 Monitoring and control systems

3.6.1 Overview of monitoring and control systems

- **Monitoring system:** a system designed to 'watch' or monitor some state external to the computer system
- **Control system:** a system designed to manage, command, direct or regulate the behaviour of other devices or systems.
  - **Event-driven system:** the controller alters the state of the system in response to some event.
  - **Time-driven system:** the controller takes action at a specific point in time or after a certain time has lapsed.

{S15-P31} **Question 5:**

A gardener grows vegetables in a greenhouse. For the vegetables to grow well, the temperature needs to always be within a particular range.

The gardener is not sure about the actual temperatures in the greenhouse during the growing season.

The gardener installs some equipment. This records the temperature every hour during the growing season.

(a) Name the type of system described. [1]

(b) Identify **three** items of hardware that would be needed to acquire and record the temperature data. Justify your choice for each. [6]

Solution:

(a) Monitoring system [1]
(b) General format:
Hardware name [1]
Hardware purpose [1]
3x

Examples:
temperature sensor
transmits measured temperature.
analogue to digital converter
converts analogue signal from sensor to digital value that can be stored.
storage device // data logger
for recording readings from sensor.
transmission hardware
to transfer data from sensor to storage device
processor
to process incoming data.

- **Hardware requirements:**
  - **Transducer:** a device which converts one form of energy to another.
  - **Sensor:** the controller alters the state of the system in response to some event.
  - **Processor:** the controller takes action at a specific point in time or after a certain time has lapsed.
  - **Actuator:** the controller alters the state of the system in response to some event.

{S17-P31} **Question 6:**

A computer system is used to manage some of the functions in a vehicle. The vehicle has a number of sensors and actuators. One sensor is used to monitor the moisture on the screen. If the moisture exceeds a pre-set value, the windscreen wiper motor turns on automatically.

The software used in the computer system is dedicated to the sensor management functions.

When the system starts, the software runs some initial tasks. It then loops continuously until the system is switched off.

(a)

(i) State the name given to the type of system described. [1]

(ii) Explain your answer to **part (i)**. [1]

(b) Within the software loop, the value of each sensor is read in turn. The value read from the sensor is then processed.

State **two** drawbacks with this method of reading and processing sensor data.

[2]

Solution:

(a) (i) Control system

(ii) Use of actuators means that the system is controlling

(b) System wastes processor time checking for values that are not changing. [1]

Some sensor input needs to be acted upon immediately. [1]

- Show understanding of the software requirements of these systems. <Bulleted>
- **Real time programming:** A structured method of programming which continually runs for the period for which the system is switched on.
- This is usually achieved by a time delay inside a loop or a sequence for reading values.
- **Feedback:** the return of a fraction of the output signal from a device to the input of the same device.
- Feedback in turn allows the system to take the output into consideration allowing the system to tweaks its performance to meet the desired output.

3.6.2 Bit manipulation to monitor and control devices

- Each bit is used to represent an individual flag.
- Therefore by altering the bits flags could be operated upon.
- **Bit manipulation operations:**
- **Masking:** an operation that enables defines which bits you want to keep, and which bits you want to clear.
  - **Masking to 1:** The OR operation is used with a 1.
  - **Masking to 0:** The OR operation is used with a 0.
- **Matching:** an operation that allows the accumulator to compare the value it contains to the given value in order to change the state of the status register.
- **Practical applications:**
  - **Setting an individual bit position:**
    - Mask the content of the register with a mask pattern which has 0 in the 'mask out' positions and 1 in the 'retain' positions.
    - Set the the result with the match pattern by using the AND command with a direct address.
  - **Testing one or more bits:**
    - Mask the content of the register with a mask pattern which has 0 in the 'mask out' positions and 1 in the 'retain' positions.
    - Compare the result with the match pattern by using the CMP command or by "Checking the pattern".
  - **Checking the pattern**
    - Use AND operation to mask out the bits and obtain a resultant.
    - Now subtract the matching bit pattern from the resultant.
    - The final 'non-zero' result confirms the patterns are not the same else vice versa.

{S15-P33} **Question 6:**

- Each greenhouse has eight sensors (numbered 1–8).
- The byte at address 150 is used to store eight 1-bit flags.
- A flag is set to indicate whether its associated sensor reading is waiting to be processed.
- More than one sensor reading may be waiting to be processed at any particular moment.
- Data received from the sensors is stored in a block of eight consecutive bytes (addresses 201–208).
- The data from sensor 1 is at address 201, the data from sensor 2 is at address 202, and so on.

Sensor number

	1	2	3	4	5	6	7	8
150	0	1	0	0	0	1	0	1

Sensor number

	1	2	3	4	5	6	7	8
150	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**

(d)

- (ii) The accumulator is loaded with the data from location 150.
- Write the assembly language instruction to check whether there is a value waiting to be processed for sensor 6.

LDD 150 // data loaded from address 150

[3]

Solution:

(d)

(ii) In order to check the value for sensor 6 we must only take the 6th bit of the accumulator into consideration. For this we must first mask out the other bits. [1]

00000100_2

(Or)

04_{16}

(Or)

4_{10}

Next, we need to check the byte, we do this by using the AND operation which directly addresses the previous byte. [1]

AND #n

In order to indicate which number system to the processor we add a symbol that represents the respective number system before the byte.
[1]

Number system	Pre-Fix
Binary	B
Hexadecimal	&
Denary	None

Thus, obtaining the final answer
AND #B000000100
(Or)
AND #&04
(Or)
AND #4

CIE A2-LEVEL COMPUTER SCIENCE//9608

WWW.ZNOTES.ORG