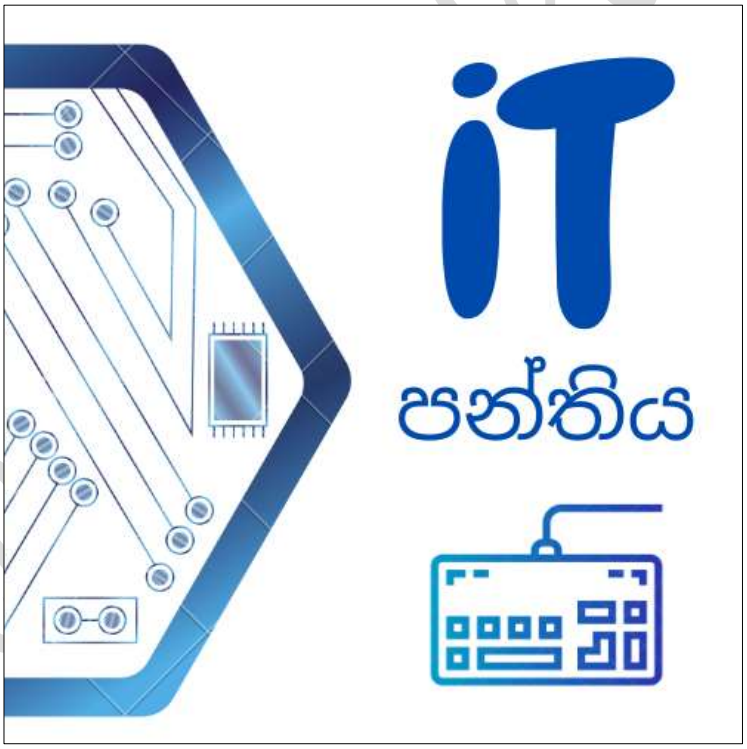
2025 MODEL PAPER 03

Paper II - Analyzation



Structured Essay

Q1 -

(a) (i) Two services provided by cloud computing

1. Infrastructure as a Service (IaaS)
2. Software as a Service (SaaS) ***(02 marks)***

(ii) Three steps of the FETCH–EXECUTE cycle of a computer

1. **Fetch:** The control unit retrieves the next instruction from main memory (RAM) into the instruction register.
2. **Decode:** The control unit interprets/decodes the fetched instruction to determine what action is required.
3. **Execute:** The CPU carries out the instruction, e.g. performing a calculation or moving data.

(03 marks)

(b) Match each sentence to the most suitable OSI layer

(i) Session Layer
(ii) Data Link Layer
(iii) Network Layer
(iv) Presentation Layer
(v) Physical Layer ***(05 marks)***

Q2 –

(a) (i) A process can move from **Blocked → Ready (Suspended)** when the event it was waiting for completes (e.g., I/O finishes) **but there isn't enough main memory** to bring it into Ready; so it's kept swapped out on disk in the Ready-Suspended state. ***(02 marks)***

(ii) Running → Terminated is irreversible because the OS releases the process's resources and destroys its control information (PCB, address space, open files, etc.). With that state gone, the process can't resume; it would have to be created anew. ***(02 marks)***

(b) FAT excerpt (block size = 8 KB).
Chain for the file starting at 400 is: 400 → 401 → 405 → 402 → 404 → end.

(i) The directory entry stores the **starting block (first cluster)**.
Answer: **400**. ***(01 mark)***

(ii) File grows by **16 KB = 2 blocks**. Allocate two free blocks and extend the chain:

- Change FAT[404] to point to the first new block (say **B1**),
- Set **FAT[B1] = B2**,
- Set **FAT[B2] = −1** (end of file).
  (Any actual free block numbers can be used.) ***(01 mark)***

(c) Given: 24-bit virtual addresses, 8 MB physical memory, page size 2 KB.

(i) Offset bits = $\log_2$(2 KB) = **11** → Page-number bits = 24 − 11 = **13 bits**. ***(01 mark)***

(ii) Frames in physical memory = 8 MB / 2 KB = 8,388,608 / 2,048 = **4,096 = 2^{12} → 12 bits** for the **frame number**. ***(01 mark)***

(d) (i) For **P1: Running → Blocked**
Update its PCB:

- **State** set to *Blocked*.
- Save **Program Counter** (next instruction).
- Save **CPU registers** (R1, R2, etc.).
- Record/update **I/O wait info** (waiting for *Printer*).
  (Other fields like priority remain the same.) ***(01 mark)***

(ii) For **P3: Blocked → Ready** (keyboard I/O completed)
Update its PCB:

- **State** set to *Ready*.
- Clear/mark completion of **I/O wait** (no longer waiting for keyboard).
- Keep PC and registers as saved; place process in the ready queue (priority unchanged). ***(01 mark)***

Q3 –

(a) (i) Current normal form: 3NF (also BCNF).
All attributes (Name, Department, Phone) are single-valued and fully dependent on the key **LecturerID**; there are no partial or transitive dependencies. ***(01 mark)***

(ii) Next normal form: Already in BCNF → **no changes needed.** ***(01 mark)***

(b) (i) Students not enrolled in any course

```
SELECT Name
FROM Student
WHERE CourseID IS NULL;
```
(01 mark)

(ii) Lecturers who teach at least two courses (name & department)

```
SELECT l.Name, l.Department
FROM Lecturer AS l
JOIN Course AS c ON c.LecturerID = l.LecturerID
GROUP BY l.LecturerID, l.Name, l.Department
HAVING COUNT(*) >= 2;
```
(01 mark)

(iii) Students who share the same course as "Jane Doe" (name & phone)

```
SELECT s2.Name, s2.Phon
FROM Student AS s1
JOIN Student AS s2 ON s2.CourseID = s1.CourseID
WHERE s1.Name = 'Jane Doe'
  AND s2.Name <> 'Jane Doe';
```
(02 marks)

(iv) Students with their lecturers' names for courses taught by lecturers in "Data Analytics"

```
SELECT s.Name AS StudentName, l.Name AS LecturerName
FROM Student AS s
JOIN Course  AS c ON s.CourseID = c.CourseID
JOIN Lecturer AS l ON c.LecturerID = l.LecturerID
WHERE l.Department = 'Data Analytics';
```
(02 marks)

(v) Students whose phone ends with "34" and are enrolled in "Computer Science"

```
SELECT s.Name
FROM Student AS s
JOIN Course AS c ON s.CourseID = c.CourseID
WHERE s.Phone LIKE '%34'
  AND c.CourseName = 'Computer Science';
```
(02 marks)

Q4 –

(a) (i)

- **One's complement:** Negative numbers are obtained by inverting all bits.
  Example: +5 (8-bit) = `00000101`; −5 = `11111010`.
- **Two's complement:** Invert all bits and add 1.
  Example: +5 = `00000101`; −5 = `11111011`. ***(01 mark)***

(ii) Two's complement has a single zero (no "−0") and addition/subtraction use the same binary adder—simpler hardware. ***(01 mark)***

(b) (i) One's complement of `01101010` → **`10010101`**. ***(01 mark)***

(ii) Subtract `10110110 - 01101001` using one's complement:

1. Take one's complement of subtrahend: `01101001` → `10010110`.
2. Add to minuend: `10110110 + 10010110 = 1 01001100` (carry out = 1).
3. Add carry back: `01001100 + 1 = 01001101`.
   Result: **`01001101`**. ***(01 mark)***

(c) (i) Two's complement of `01011011`:
Invert → `10100100`, add 1 → **`10100101`**. ***(01 mark)***

(ii) Add `11010101 + 00111001`:

```
  11010101
+ 00111001
-----------
  100011110
```

8-bit result = **`00011110`** (carry out 1 ignored).
No overflow: signs of addends differ or carry into/out of sign bit matches. ***(01 mark)***

(d) (i) Two's complement for −45 (8-bit):
+45 = `00101101`
Invert → `11010010`
Add 1 → **`11010011`**. ***(01 mark)***

(ii) Two's complement lets the same adder handle +/− numbers; no separate subtraction circuits—simpler hardware implementation. ***(01 mark)***

(e)

i. **non-volatile memory**
ii. **instruction pipelining**
iii. **hyper-threading**
iv. **virtua l memory**
v. **static RAM (SRAM)**
vi. **memory bandwidth** ***(02 marks)***

Essay Type

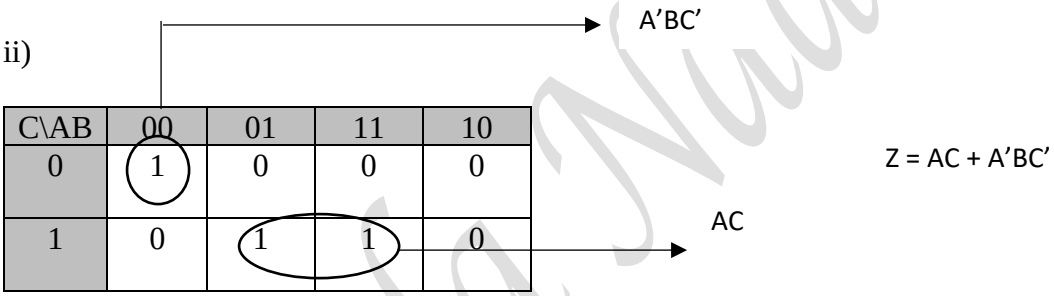
Q1 –

- (a) i) Z=(A'·B)+(A·C') ***(01 mark)***

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

b) i) (A+B')(A+C)(B+C') ***(02 marks)***
=AA+AC+AB'+B'C (B+C')
=[A (1+C+B')+ B'C] (B+C')
= AB + BB'C + AC' + B'CC'
= AB + AC'

ii) ***(01 mark)***



C\AB	00	01	11	10
0	1	0	0	0
1	0	1	1	0

Z = AC + A'BC'

(c) i) ***(01 mark)***

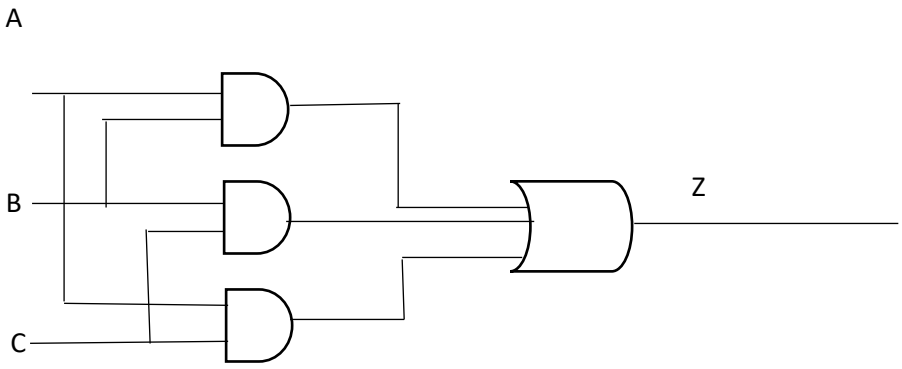
A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

ii) ***(01 mark)***

C\AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

iii) Z = AB + AC + BC *(01 mark)*

iv) *(03 marks)*



(d) i) A **full adder** is a combinational circuit that adds three binary inputs: *(01 mark)*

- Two significant bits (A and B)
- A carry-in bit (Cin) from a previous addition
  It produces two outputs:
- **Sum (S)**
- **Carry-out (Cout)**

• Difference from a **half adder**: *(01 mark)*

- A **half adder** adds only two inputs (A and B) and gives a Sum and Carry, but it does **not** consider a carry input from a previous stage.
- A **full adder** includes the carry-in, making it suitable for multi-bit binary addition.

ii) *(01 mark)*

A	B	SUM	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

iii)• A **flip-flop** is a basic memory element in digital electronics. It can store **1 bit of information** (either 0 or 1). Its state changes only at specific clock edges (synchronous behavior), making it essential in registers, counters, and sequential circuits. *(01 mark)*

• Difference from **combinational logic gates**: *(01 mark)*

- **Combinational logic**: Output depends only on current inputs (no memory).
- **Sequential logic (flip-flops)**: Output depends on current inputs **and previous state** (has memory).

Q2 –

(a) Diagram (text): ***(01 mark)***

```
[PC-A NIC] <==== crossover UTP (Cat5e/Cat6) ====> [PC-B NIC]
```

- A **crossover Ethernet cable** swaps the TX/RX pairs so each NIC's **TX pins (1,2)** go to the other NIC's **RX pins (3,6)** and vice-versa.
- Each PC auto-negotiates speed/duplex and communicates directly (no switch).

(b) (i) Topology (text): ***(04 marks)***

```
          Internet
             |
         [ Router ]
          /      \
   [Switch HR]   [Switch FIN]
    |  |  |  | \            \
  H1 H2 H3 H4 H5             F1 F2 F3 F4 F5
    |   \
  [FS] [PR]
```

- HR PCs (H1–H5), **File Server (FS)** and **Printer (PR)** plug into **Switch HR**.
- Finance PCs (F1–F5) plug into **Switch FIN**.
- Both switches uplink to the **Router**, which also connects to the Internet.

(ii) Why a router & switches + placement ***(01 mark)***

- **Switches** (one per LAN) provide local connectivity, MAC-based forwarding, one collision domain per port.
- The **Router** interconnects the HR and Finance subnets, performs **routing** between them, and provides **Internet access** (NAT, firewall, DHCP). It sits **between the two switches and the ISP**.

(iii) F3 sends a document to PR (data flow) ***(01 mark)***

1. F3 sees PR is on a different subnet → sends the packet to its **default gateway (router)** (after ARP for router's MAC).
2. Router routes the packet to the **HR subnet** and forwards it out the HR interface (after ARP for PR's MAC).
3. **Switch HR** delivers the frame to **PR**, which receives/prints the job. (Return acknowledgements go back the reverse path.)

(c) (i) Subnet mask: `/26` → **255.255.255.192** ***(01 mark)***

(ii) Four /26 subnets ***(02 marks)***

Subnet	Network address	1st usable	Last usable	Broadcast
S1	172.16.0.0/26	172.16.0.1	172.16.0.62	172.16.0.63
S2	172.16.0.64/26	172.16.0.65	172.16.0.126	172.16.0.127
S3	172.16.0.128/26	172.16.0.129	172.16.0.190	172.16.0.191

Subnet	Network address	1st usable	Last usable	Broadcast
S4	172.16.0.192/26	172.16.0.193	172.16.0.254	172.16.0.255

(Each has 64 addresses, **62 usable**.)

(d) (i) Role when typing www.example.com ***(01 mark)***

- Your host asks a **recursive resolver** (usually ISP/OS), which queries (often iteratively) the **root**, then the **TLD (.com)**, then the **authoritative** nameserver to obtain the **IP address**, caches it, and returns it so the browser can connect.

(ii) How the hierarchical DNS speeds/scales ***(01 mark)***

- **Delegation** from root → TLD → authoritative spreads the load.
- **Distribution** of zones and **caching** at each level reduce query time and improve scalability and resilience.

(e) ***(02 marks)***

i) End-to-end delivery & reliability → **Transport layer** (TCP).
ii) Packet formatting + source/destination IPs → **Internet layer** (IP).
iii) Send to next hop/physical network → **Network Access (Link) layer**.
iv) Manage interaction/session parameters between app & transport → **Application layer** (closest TCP/IP equivalent to OSI Session).

(f) ***(01 mark)***

(i) Encryption key used: 3 (shift of 3).
(ii) Encrypt "HELLO": KHOOR.
(iii) Alice's encrypted response for "WORLD": ZRUOG.

Q3 –

(a) ***(04 marks)***

i. **Cloud computing**
ii. **Distributed computing**
iii. **RAID**
iv. **Magnetic tape**
v. **SSD**
vi. **DRAM**
vii. **ROM**
viii. **Optical disk**

(b) ***(02 marks)***

Characteristic	Label	Answer
Stores data sequentially and is used for backups	A	**Magnetic tape**

Characteristic	Label	Answer
Uses laser technology for reading/writing data	B	**Optical disk**
High-speed memory used in cache systems	C	**Cache memory**
Provides permanent storage for multimedia files	D	**Hard disk**

(c) (i) SSD vs. traditional hard drives *(03 marks)*

- **Advantage:** Much faster read/write speeds → quicker boot times and file access.
- **Limitation:** Higher cost per gigabyte and limited write endurance compared to HDDs.

(ii) DRAM vs. ROM *(03 marks)*

Feature	DRAM	ROM
Speed	Very fast, used as main system memory.	Slower, used mainly for firmware.
Volatility	Volatile – contents lost when power off	Non-volatile – retains data without power.
Example	PC main memory (e.g., 8 GB DDR4)	BIOS/UEFI firmware chip.

(iii) Distributed computing *(03 marks)*

- **Explanation:** Breaks a large computational problem into smaller tasks that run simultaneously on many networked computers, combining results for faster overall solution.
- **Real-world example: SETI@home** project or **Google MapReduce/Hadoop** clusters for big-data analysis.

Q4 –

(a) (i) *(02 marks)*
1st instruction (given): `LOAD base → R1`
2nd: `LOAD height → R2`
3rd: `MUL R1, R2 → R3` (form the product)
4th: `MOVE R3 → ACC/Result-reg` (prepare to store)
5th (given): `STORE ACC → result`

(ii) Subtract $11011_2 - 10110_2$ by 2's-complement addition. *(02 marks)*
2's complement of 10110_2 → invert 01001_2, add 1 → 01010_2.
Now add: $11011_2 + 01010_2 = 100101_2$.
Ignore the carry → 00101_2, which equals $11011_2 - 10110_2$.

(b) (i) The preempted **text editor** goes from **RUNNING → READY**. *(01 mark)*
(ii) The web browser waiting for network data goes **RUNNING → BLOCKED** (and later **BLOCKED → READY** when data arrives). *(01 mark)*
(iii) On a context switch, the **Program Counter (PC)** in each process's PCB is saved/restored so each process resumes **exactly at the next instruction** where it was stopped. *(01 mark)*

(c) (i) Frames in physical memory = 64 KB/2 KB== 32 **frames**. ***(01 mark)***

(ii) Page size 2 KB → **offset = 11 bits**. ***(02 marks)***
Virtual address `0001 0000 1000` ⇒ **page # = 1**, **offset = 0000 1000**.
From the page table, page 1 maps to **frame 0011**.
Physical address = frame (0011) || offset (0000 1000) ⇒ `0011 0000 1000`**.**

(iii) The OS will not map a virtual address whose page has **Validity Bit = 0** (page not resident), so the reference causes a **page fault**; the page must be brought into memory before mapping. ***(01 mark)***

(iv) A **Modified (dirty) Bit** marks pages that have been **written to**. On replacement, only dirty pages are written back to disk, avoiding unnecessary writes and improving performance. ***(01 mark)***

(d) (i) For **indexed allocation**, the OS needs the **address of the index block** for *report.txt*. That index block contains the list of data block numbers (e.g., 50, 60, 70, 80) used by the file. ***(01 mark)***

(ii) With **contiguous allocation**, a file must occupy one continuous run of blocks. Over time free space becomes scattered; a large enough **contiguous** run may not exist even if the total free space is sufficient—this is **external fragmentation**. Indexed allocation does not require contiguity, so it avoids this problem. ***(02 marks)***

Q5 –

(a) (i) Current normal forms

- **Book(Book_ID, Title, Author, Publisher, ISBN) → 2NF (not 3NF)**.
  Reason: `Book_ID → ISBN` and `ISBN → {Title, Author, Publisher}` → transitive dependency on the key.
- **Member(Member_ID, Member_Name, Address, Phone_No) → 3NF/BCNF**.
  All non-keys depend only on `Member_ID`.
- **Loan(Loan_ID, Member_ID, Book_ID, Loan_Date, Return_Date) → 3NF/BCNF**.
  `Loan_ID` is the key; other attributes depend only on it. ***(02 marks)***

(ii) Convert each to the **next** normal form

- **Book → 3NF decomposition**
  - **Books**(`ISBN` PK, Title, Author, Publisher)
  - **Copies**(`Book_ID` PK, `ISBN` FK)
- **Member** already 3NF → **BCNF** (no change):
  - **Member**(`Member_ID` PK, Member_Name, Address, Phone_No)
- **Loan** already 3NF → **BCNF** (no change):
  - **Loan**(`Loan_ID` PK, `Member_ID` FK, `Book_ID` FK, Loan_Date, Return_Date) ***(02 marks)***

(If you must fill the table with placeholders P…U, map for example: P=3NF, S=Books+Copies; Q=BCNF, T=Member; R=BCNF, U=Loan.)

(b) ER diagram (described)

- **Entities & PKs**
  - **Book**(ISBN)
  - **Copy**(Book_ID, ISBN FK→Book)
  - **Member**(Member_ID)
  - **Loan**(Loan_ID, Member_ID FK→Member, Book_ID FK→Copy)
- **Relationships & cardinalities**

- Member **borrows** Copy **via** Loan:
  Member (1) —< Loan >— (1) Copy.
  A member can have many loans; a copy can appear in many loans (over time). ***(02 marks)***

(c) Overdue loans (Return_Date not updated and Loan_Date > 14 days ago) ***(01 mark)***

(i) List overdue loans with member name: ***(02 marks)***

```
SELECT m.Member_Name, l.Book_ID, l.Loan_Date, l.Return_Date
FROM Loan l
JOIN Member m ON m.Member_ID = l.Member_ID
WHERE l.Return_Date IS NULL
  AND l.Loan_Date <= CURRENT_DATE - INTERVAL '14' DAY;
```

(ii) Total number of books currently borrowed per member: ***(02 marks)***

```
SELECT m.Member_ID, m.Member_Name, COUNT(*) AS TotalBorrowed
FROM Member m
JOIN Loan   l ON l.Member_ID = m.Member_ID
WHERE l.Return_Date IS NULL
GROUP BY m.Member_ID, m.Member_Name
ORDER BY m.Member_ID;
```

(d) Reservations

(i) Table schema ***(02 marks)***

```
CREATE TABLE Reservation (
  Reservation_ID INT PRIMARY KEY,
  Member_ID      INT NOT NULL,
  Book_ID        INT NOT NULL,      -- reserve a specific copy; use ISBN if reserving a
title
  Reservation_Date DATE NOT NULL,
  FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID),
  FOREIGN KEY (Book_ID)   REFERENCES Copies(Book_ID)
);
```

(ii) Reservations made by members who have never borrowed any book ***(02 marks)***

```
SELECT r.*
FROM Reservation r
JOIN Member m ON m.Member_ID = r.Member_ID
LEFT JOIN Loan   l ON l.Member_ID = m.Member_ID
WHERE l.Member_ID IS NULL;
```

Q6 –

(a) **(i) Fragmentation** ***(02 marks)***

- *Fragmentation* is wasted/unusable space that occurs when memory/storage is allocated and freed over time.
- **Internal fragmentation:** Allocated block is larger than the requested size; the **unused space inside** the block is wasted.
- **External fragmentation:** Free space exists but is **split into many small non-contiguous holes**, so a large request cannot be satisfied even though total free space is enough.

(ii) Thrashing (virtual memory) *(01 mark)*
Excessive page faults when a process's working set doesn't fit in RAM (too few frames, frequent switching). The CPU spends most time swapping pages to/from disk, causing **severe performance degradation**.

(iii) DMA (Direct Memory Access) *(01 mark)*
A DMA controller lets I/O devices transfer data **directly to/from main memory** without the CPU moving each byte. CPU is interrupted only on completion or in large chunks, **reducing CPU overhead** and bus traffic, improving throughput.

(b) (i) Wasted space in last cluster *(02 marks)*
File size = 18,400 B; cluster = 1,024 B.
Clusters needed = ⌈18400 / 1024⌉ = 18 → allocated = 18 × 1024 = 18,432 B.
Wasted space = 18,432 − 18,400 = 32 bytes.

(ii) Linked file allocation (how retrieval works & a drawback) *(01 mark)*
The file's directory entry stores the **first block**. Each block contains a **pointer to the next block** (or FAT entry points to the next). To read the file, the OS **follows the chain** block by block until the end marker.
Drawback: Slow random access (must traverse the chain), and corruption of a single pointer can lose the remainder of the file.

(c) (i) Contiguous: *(01 mark)*

- Each process occupies **one continuous block** in memory.
- **Simple and fast** address translation (base + limit).
- Suffers **external fragmentation**; compaction may be needed.

(ii) Segmented: *(01 mark)*

- Program divided into **logical segments** (code/data/stack) of variable length.
- Uses a **segment table** with base & limit; supports **protection and sharing** at segment level.
- Still prone to **external fragmentation**.

(iii) Paging: *(01 mark)*

- Memory split into **fixed-size pages/frames**; page table maps virtual pages to frames.
- **No external fragmentation**; allows non-contiguous physical memory.
- Possible **internal fragmentation** in the last page; requires page-table overhead.

(d) IEEE-754 single-precision for 37.625

1. Decimal → binary: $37.625_{10} = \mathbf{100101.101_2}$.
2. Normalize: $\mathbf{1.00101101_2 \times 2^5}$.
3. Sign bit **0** (positive).
4. Exponent = 5 + bias(127) = $\mathbf{132 = 10000100_2}$.
5. Fraction (mantissa) = bits after the leading 1: **00101101000000000000000** (pad to 23 bits).

Final 32-bit form:
`0 10000100 00101101000000000000000` *(03 marks)*

(e) Match devices to usage

- **1. Barcode Scanner → (i)** Captures printed barcodes for product identification.

- **2. Microphone → (ii)** Used to record audio input.
- **3. LED Monitor → (iii)** Displays high-resolution output.
- **4. Blu-ray Disk → (iv)** Stores large multimedia files. ***(02 marks)***
-