Fundamental Of Computer Organization

Assignments

Ch.1 NUMBER SYSTEM AND LOGIC GATES

1. Explain different Number Systems with examples.

A number system is a method of representing numbers using digits or symbols. Different number systems are used in computers because computers understand only binary signals.

(1) Decimal Number System (Base 10)

- Digits: **0–9**
- Most commonly used in daily life.

Example:
$245_{10} = (2 \times 10^2) + (4 \times 10^1) + (5 \times 10^0)$

(2) Binary Number System (Base 2)

- Digits: **0 and 1 only**
- Computers use this system because electronic circuits work on ON/OFF (1/0) signals.

Example:
$1101_2 = 1\times8 + 1\times4 + 0\times2 + 1\times1 = 13$

(3) Octal Number System (Base 8)

- Digits: **0–7**
- Used as a shortcut for binary.

Example:
$57_8 = 5\times8 + 7\times1 = 47$

(4) Hexadecimal Number System (Base 16)

- Digits: **0–9 and A–F**
  (A=10, B=11, C=12, D=13, E=14, F=15)

Example:
$2F_{16} = 2\times16 + 15 = 47$

2. Converting Decimal to Binary, Octal, Hexadecimal

(A) Decimal to Binary

Use **division by 2** and write remainders.

Example: Convert 25 to Binary

25 ÷ 2 = 12 R1
12 ÷ 2 = 6 R0
6 ÷ 2 = 3 R0
3 ÷ 2 = 1 R1
1 ÷ 2 = 0 R1

Binary = **11001_2**

(B) Decimal to Octal

Divide by **8**.

Example: Convert 125

125 ÷ 8 = 15 R5
15 ÷ 8 = 1 R7
1 ÷ 8 = 0 R1

Octal = **175_8**

(C) Decimal to Hexadecimal

Divide by **16**.

Example: Convert 254

254 ÷ 16 = 15 R14 → E
15 ÷ 16 = 0 R15 → F

Hex = **FE_{16}**

3. Perform Binary Addition & Subtraction

(A) Binary Addition Rules

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 (carry 1)

Example

```
  1011
+ 1101
--------
 11000
```

(B) Binary Subtraction Rules

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1 (borrow 1)

Example

```
  1010
-  0111
--------
   0011
```

4. Complements and Use to Represent Negative Numbers

Complements are used for simplifying subtraction and representing negative numbers.

(A) 1’s Complement

- Flip all bits (0→1, 1→0)

Example:
1’s complement of 1010 = **0101**

(B) 2’s Complement

- 1’s complement + 1

Example:
Number = 0101 (5)
1’s complement = 1010
+1 = **1011** (represents −5)

Why Complements?

- Used to represent **negative numbers**
- Simplifies subtraction
- CPU uses **2’s complement** for all arithmetic

5. What is a Logic Gate? Explain AND, OR, NOT with Truth Table & Symbols.

Logic Gates are digital circuits that perform logical operations on binary inputs.

(A) AND Gate

- Output is 1 only if **both** inputs are 1.

Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

(B) OR Gate

- Output is 1 if **any one** input is 1.

Truth Table

A	B	Y
0	0	0

A	B	Y
0	1	1
1	0	1
1	1	1

(C) NOT Gate

- Also called **Inverter**
- Output is opposite of input.

Truth Table

A	Y
0	1
1	0

6. Evaluation of Logical Expressions Using Basic Gates (7 Marks)

Logical expression uses AND, OR, NOT to produce results.

Example Expression:

$Y = A + (B \cdot C)$

With Parentheses:

1. First evaluate **$B \cdot C$** using AND
2. Add result to A using OR

Example Calculation

A=1, B=1, C=0

Step 1:
$B \cdot C = 1 \cdot 0 = 0$

Step 2:
$A + 0 = 1$

→ **Y = 1**

Without Parentheses

Follow operator precedence:

1. NOT
2. AND
3. OR

Example:
Y = A + B · NOT C

Compute C’, then B·C’, then OR with A.

7. NAND & NOR Gates + Universal Gate Property (7 Marks)

(A) NAND Gate

- Output is 0 only when both inputs are 1.
- NAND = NOT(AND)

Truth Table

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(B) NOR Gate

- Output is 1 only when both inputs are 0.
- NOR = NOT(OR)

Truth Table

A	B	Y
0	0	1
0	1	0

A	B	Y
1	0	0
1	1	0

Universal Gate Property

Both **NAND** and **NOR** can be used to build:

✔ AND
✔ OR
✔ NOT
✔ XOR
✔ All digital circuits

→ That is why they are called **UNIVERSAL GATES**.

8. Perform Conversions + Binary Arithmetic + Logic Evaluation (5–7 Marks)

(A) Number System Conversions

Binary ↔ Octal

- Group bits in 3
  Binary ↔ Hex
- Group bits in 4

Example:
Binary 110111 → Hex
= 0110 1111 → $6F_{16}$

(B) Binary Arithmetic

Example:

```
 1010
+0111
-----
 10001
```

(C) Logic Expression Evaluation

Expression:
$$Y = A \cdot B + C'$$

A=1, B=0, C=1
$C' = 0$
$A \cdot B = 0$
$0 + 0 = 0$

Y = **0**

Ch.2 BASIC STRUCTURE OF COMPUTERS

1. Explain different types of computers with examples. (7 Marks)

Computers can be classified based on **size**, **purpose**, and **data handling**.

A. Based on Size

(1) Supercomputer

- Most powerful & fastest computers.
- Used for complex scientific simulations.
- **Examples:** PARAM, Summit, Fugaku.

Applications: Weather forecasting, nuclear research.

(2) Mainframe Computer

- Support thousands of users at once.
- Very high storage and processing capacity.
- **Examples:** IBM Z-series.

Applications: Banking, railways, large enterprises.

(3) Minicomputer

- Mid-range computers for medium-sized organizations.
- Multi-user support.
- **Examples:** PDP-11, IBM AS/400.

(4) Microcomputer

- Smallest and most commonly used.
- **Examples:** Desktop, laptop, tablet.

B. Based on Purpose

(1) General Purpose Computers

- Used for general tasks: office work, browsing.
- **Examples:** PCs, laptops.

(2) Special Purpose Computers

- Designed for a single task.
- **Examples:** ATM, washing machine controller.

C. Based on Data Handling

(1) Digital Computers

- Work with binary data (0,1).
- **Example:** All modern PCs.

(2) Analog Computers

- Work with continuous values.
- **Example:** Speedometer, thermometer.

(3) Hybrid Computers

- Combination of Analog + Digital.
- **Example:** ICU monitoring systems.

2. Functional Units of a Computer with Block Diagram

A computer consists of 5 major functional units:

```
+------------------+
|    Input Unit    |
+--------+---------+
```

```
                          |
                          v
+-------------+      +------------------+      +-------------+
|  Memory     |<---->|   CPU            |<---->|  Output     |
|  Unit       |      | (ALU + CU  |      |  Unit       |
+-------------+      +------------------+      +-------------+
```

(1) Input Unit

- Accepts data and instructions.
- **Examples:** Keyboard, mouse, scanner.

(2) Memory Unit

(a) Primary Memory

- RAM, ROM
- Fast but limited capacity.

(b) Secondary Memory

- Hard Disk, SSD, Pen drive
- Large storage capacity.

(3) Central Processing Unit (CPU)

(a) ALU (Arithmetic Logic Unit)

- Performs arithmetic operations (add, subtract)
- Performs logic operations (AND, OR)

(b) CU (Control Unit)

- Controls all operations.
- Sends signals for instruction execution.

(4) Output Unit

- Displays results.
- **Examples:** Monitor, printer.

(5) I/O Interfaces

- Allow devices to communicate with CPU.

3. Basic Operational Concepts of a Computer

Computer operations follow the **stored program concept**.

Steps:

1. **Fetch** the instruction from memory
2. **Decode** the instruction
3. **Execute** the operation
4. **Store** the output

Example

For instruction:
ADD R1, R2

1. Fetch ADD instruction from memory
2. Decode as "Add R1 and R2"
3. ALU adds values
4. Result stored in R1

4. What is Bus Structure? Types of Buses

A **Bus** is a communication pathway that transfers data between components.

Types of Buses

(1) Data Bus

- Carries **actual data**.
- Bi-directional.

(2) Address Bus

- Carries addresses of memory locations.
- Uni-directional.

(3) Control Bus

- Sends control signals (Read, Write, Interrupt).
- Bi-directional.

(4) System Bus

- Combination of Data + Address + Control buses.

(5) PCI / USB / SATA Buses

Modern standards for fast communication.

5. Difference Between Multiprocessors and Multicomputer

Feature	Multiprocessors	Multicomputer
Definition	Multiple processors in **one** system	Many independent computers connected by network
Memory	Shared memory	Distributed memory
Communication	Shared bus	Message passing
Failure	One fails → system still works	One node fails → affects network
Speed	Very fast	Slower than multiprocessors
Example	Dual-core, Quad-core CPUs	Cluster computers

Advantages of Multiprocessors

- Faster performance
- Shared memory → easier communication

Disadvantages

- Expensive
- Complex design

Advantages of Multicomputers

- Low cost
- Highly scalable

Disadvantages

- Slower communication
- Harder programming model

6. Data Representation Techniques

Computers represent data in binary form.

Types of Data Representation

1. **Binary**
2. **Octal**
3. **Hexadecimal**
4. **ASCII / Unicode**
5. **Signed Numbers (using complements)**

Fixed Point Representation

Used for **integers**.

Example

Binary: 11001 = 25

Advantages

- Simple and fast
- Needed for integer arithmetic

Disadvantages

- Cannot represent fractions

Floating Point Representation

Represents **real numbers** in scientific form:

Number = Mantissa × 2^Exponent

Example

1.101×2^3

Advantages

- Represents very large & very small numbers
- High precision

Disadvantages

- Slow
- Complex hardware

7. Error Detection Codes

Used to detect errors in data transmission or storage.

(1) Parity Bit

- Adds a bit to make number of 1s **even** (even parity) or **odd** (odd parity).

Example

Data: 1011
Even parity → add 0 → 10110

(2) Checksum

- Sum of data blocks → transmitted along with data.
- Receiver recomputes & compares.

(3) Cyclic Redundancy Check (CRC)

- Uses polynomial division.
- Highly accurate; used in networking & storage.

(4) Hamming Code

- Detects & corrects **single-bit errors**.

Example:

Creates parity bits at positions 1,2,4,8…

8. Identify types of computers, explain operations & performance factors (7 Marks)

A. Types of Computers

- Supercomputer
- Mainframe
- Minicomputer
- Microcomputer
- Analog
- Digital
- Hybrid

B. Components of Computer

- CPU
- Memory
- Input/Output units

- System bus

C. Operational Concepts

- Fetch, Decode, Execute cycle
- Stored program concept (Von Neumann Architecture)

D. Factors Influencing Performance

1. Clock speed
2. Cache size
3. Number of cores
4. Memory bandwidth
5. Instruction set architecture
6. Throughput & latency

Ch.3 REGISTER TRANSFER LANGUAGE AND MICROOPERATIONS

1. Define Register Transfer Language (RTL).

Register Transfer Language (RTL) is a symbolic notation used to describe how data is transferred between registers and what operations are performed inside a digital computer.
It expresses micro-operations such as arithmetic, logic, shift, and data transfer operations.

Example:

$R1 \leftarrow R2 + R3$

Meaning: Add contents of R2 and R3, store the result in R1.

RTL is mainly used in:

- CPU design
- Micro-operations description
- Control unit implementation

2. List any four types of computer registers.

1. **Program Counter (PC):** Holds address of next instruction.
2. **Instruction Register (IR):** Stores current instruction being executed.
3. **Memory Address Register (MAR):** Holds the address to be accessed in memory.
4. **Accumulator (AC):** Stores intermediate arithmetic/logic results.

Other examples: MDR, Index Register, Temporary Register, Stack Pointer.

3. Difference between Memory-Reference Instructions and Register-Reference Instructions.

Feature	Memory-Reference Instructions	Register-Reference Instructions
Definition	Uses memory address as operand	Operates only on CPU registers
Access	Requires memory access	No memory access
Speed	Slower	Faster
Example	LOAD R1, 500	CLR AC

Example:
Memory-reference: ADD 400
Register-reference: CMA (complement AC)

4. Function of the Program Counter (PC).

The Program Counter is a special register that always holds the **address of the next instruction** to be executed.

Functions

- After fetching an instruction, PC increments automatically.
- During branch/jump operations, PC is updated with a new address.
- Controls the sequence of program execution.

5. Mention any two types of shift micro-operations. ***(5 marks)***

1. **Logical Shift (Left/Right)**
   - Vacated bits filled with 0.
2. **Arithmetic Shift (Left/Right)**
   - Preserves sign bit for signed numbers.

(OR)

3. **Circular/Rotate Shift**
    - Bits rotated around.

6. Purpose of an Instruction Cycle in a Basic Computer.

The Instruction Cycle is the sequence of steps a CPU follows to execute an instruction.

Phases:

1. **Fetch Cycle**
    - Fetch instruction from memory using PC.
    - PC = PC + 1.
2. **Decode Cycle**
    - Decode opcode to determine operation.
3. **Execute Cycle**
    - ALU performs required operation.
    - Result stored in register/memory.
4. **Interrupt Cycle**
    - Handle any interrupt requests.

Purpose:

To ensure that all instructions are executed in a systematic, controlled manner.

7. Arithmetic Micro-operations with Example.

Arithmetic micro-operations perform basic arithmetic on binary data stored in registers.

Common operations:

- Addition: $R1 \leftarrow R2 + R3$
- Subtraction: $R1 \leftarrow R2 - R3$
- Increment: $R1 \leftarrow R1 + 1$
- Decrement: $R1 \leftarrow R1 - 1$

Example:

$R1 \leftarrow R1 + R2$

ALU adds contents of R1 and R2, stores result in R1.

8. Difference between Logical Micro-operation and Arithmetic Micro-operation.

Feature	Logical Micro-operation	Arithmetic Micro-operation
Operation	Bitwise logic	Arithmetic
Functions	AND, OR, XOR, NOT	Add, Sub, Increment
Example	R1 ← R1 AND R2	R1 ← R1 + R2
Use	Masking, comparison	Computation

Logical micro-ops manipulate bits, while arithmetic micro-ops manipulate numeric values.

9. Role of the Bus in Register Transfer Operations.

A **Bus** is a common communication pathway used to transfer data between registers.

Role:

- Connects multiple registers for data movement.
- Reduces wiring complexity (only 1 set of wires needed).
- Supports operations such as R1 ← R2.
- Allows ALU, memory, and registers to communicate.

Most systems use:

- **Tri-state bus**
- **Multiplexer-based bus**

10. Format of an Instruction with Example.

An instruction format defines how bits in an instruction are divided into fields.

Common Fields:

1. **Opcode (Operation code):** Specifies operation.
2. **Address field / Operand:** Specifies register or memory location.
3. **Mode field:** Addressing mode.

Example:

Instruction: 1 011 0010 0001
Fields:
1 → Mode
011 → Opcode (ADD)
0010 → Register

0001 → Address/Immediate

11. Register Transfer Statement to Swap R1 and R2.

Use a temporary register:

TEMP ← R1
R1 ← R2
R2 ← TEMP

12. Effective Address in Relative Addressing Mode.

Formula:

Effective Address = PC + Offset

Given:
PC = 300
Offset = 25

Calculate:

EA = 300 + 25 = 325

13. Demonstrate Logical Left Shift on 4-bit Data.

Logical Left Shift moves bits left; rightmost bit becomes 0.

Example:

Data = **1011**

Shift left:

1011 → 0110

Steps:

- Left shift everything
- Insert 0 at LSB

14. Compare Input/Output Instructions and Interrupt Operations.

Input/Output (I/O) Instructions

- CPU controlled
- Used for reading/writing data from I/O devices
- Examples: IN, OUT

Characteristics

- Slower
- CPU must wait for device

Interrupt Operations

- Signals sent by devices to CPU
- CPU pauses current task
- Jumps to Interrupt Service Routine (ISR)

Characteristics

- Faster response
- CPU does not waste time waiting
- Used in keyboards, timers, disks

Comparison Table

Feature	I/O Instructions	Interrupt Operations
Method	CPU initiates I/O	Device requests CPU
Speed	Slow	Faster
CPU Waiting	Yes	No
Use	Simple transfers	Critical events

Ch.4 COMPUTER ARITHMETIC

1. Difference Between Signed-Magnitude, 1's Complement, and 2's Complement

Binary numbers can be represented in different ways to show positive and negative values.

(A) Signed-Magnitude Representation

- MSB (Most Significant Bit) = Sign bit
  - 0 → Positive
  - 1 → Negative
- Remaining bits represent magnitude.

Example:

+5 = 0 101
−5 = 1 101

Features

- Simple representation
- Two representations for zero
  - +0 = 0 000
  - −0 = 1 000

(B) One's Complement Representation

- Negative number = invert (flip) all bits.
- Positive numbers remain same.

Example:

+5 = 0101
−5 = 1010 (1's complement)

Features

- Two representations for zero (+0 and −0)
- Ending-around carry is needed during arithmetic

(C) Two's Complement Representation

- Negative number = 1's complement + 1
- Positive = same
- Only **one** representation for zero

- Widely used in computers

Example:

+5 = 0101
1's complement = 1010
+1 = 1011 → (−5)

Difference Table (for exam)

Feature	Signed Magnitude	1's Complement	2's Complement
Sign Bit	Yes	Yes	Yes
Zero Representation	Two (+0, −0)	Two	One
Negative Conversion	Change sign bit	Invert bits	Invert + add 1
Arithmetic	Complex	Complex	Simple → Used in CPUs

2. Steps for Binary Addition & Subtraction Using 2's Complement

A. Addition Using 2's Complement

Steps:

1. Represent both numbers in binary.
2. If number is negative → convert to 2's complement.
3. Add both binary numbers.
4. If carry is generated → discard it.
5. Result is final answer (if MSB is 0 = positive, if 1 = negative → convert back).

Example:
Compute 7 + (−5)

7 = 0111
−5 → 2's complement:
5 = 0101
1's = 1010
+1 = **1011**

Add:
0111
+1011
= 10010 → discard carry → **0010 = 2**

B. Subtraction Using 2’s Complement

A − B = A + (2’s complement of B)

Example:
9 − 6

9 = 1001
6 = 0110
2’s complement:
0110 → 1001 +1 → **1010**

Add:
1001
+1010
= 10011 → discard carry → **0011 = 3**

3. Overflow in Binary Arithmetic & Detection

Overflow happens when result is **too large or too small** to fit in the available number of bits.

Overflow occurs when:

- Adding two positive numbers → result becomes negative
- Adding two negative numbers → result becomes positive

Detection Rule

Check **carry into MSB** and **carry out of MSB**:

- If both carries are **different** → Overflow
- If both carries are **same** → No overflow

Example:

0111 (+7)
+0101 (+5)
=1100 (interpreted as −4 due to sign bit)

→ Positive + Positive = Negative → **Overflow**

4. Binary Multiplication Using Shift and Add

Binary multiplication is similar to decimal long multiplication.

Steps

1. Multiply multiplier LSB with multiplicand.
2. If bit is 1 → add multiplicand to result.
   If 0 → add nothing.
3. Shift multiplicand left after each step.
4. Continue for all bits.

Example: Multiply 101 × 011

```
101   (multiplicand)
011   (multiplier)
```

Step-by-step:

- LSB = 1 → Add 101
- Shift 101 left → 1010
- Next bit = 1 → Add 1010
- Shift → 10100
- Next bit = 0 → Add 0

Final addition:

```
  101
+1010
-----
 1111  = 15
```

5. Booth's Multiplication Algorithm (7 marks)

Booth's algorithm is used for signed binary multiplication using **2's complement**, making multiplication faster.

Why Booth's?

- Reduces number of additions.
- Handles positive and negative numbers.
- Efficient for sequences like 11110000.

Rules (Multiplier pair checking)

Check current bit and previous bit:

Bits (Q0,Q-1)	Action
01	Add multiplicand
10	Subtract multiplicand
00	Do nothing
11	Do nothing

Then perform **Arithmetic Right Shift**.

Benefits

- Faster than normal shift-and-add
- Useful for signed numbers
- Reduces repetition

6. Restoring Division Algorithm (7 marks)

Used for **binary division** similar to long division.

Steps

1. Left shift remainder.
2. Subtract divisor from remainder.
3. If result is positive → quotient bit = 1
4. If result is negative → restore previous remainder (add divisor back) and quotient bit = 0
5. Repeat for all bits.

Example:

Divide 1101 by 0101

A table-based explanation can be given in the exam.

Why called restoring?

Because if subtraction makes value negative, the previous value is **restored**.

7. Floating-Point Number, Format & Addition

Floating-Point Number

A number represented in scientific form:

Number = Mantissa × Base^Exponent

IEEE-754 Format (Single Precision)

Field	Bits
Sign	1
Exponent	8
Mantissa	23

Floating-Point Addition Steps

1. **Align Exponents**
   - Shift mantissa of smaller exponent.
2. **Add/Subtract Mantissas**
   - Based on sign.
3. **Normalize the Result**
   - Adjust exponent and mantissa.
4. **Round the Result**
5. **Check Overflow/Underflow**

8. Compare Fixed-Point & Floating-Point

Fixed-Point

- Decimal point at fixed position.
- Used for integers.
- Fast but limited range.

Example:
Binary: 001101 = 13

Floating-Point

- Decimal point "floats".
- Used for real numbers.
- Large range.

Example:
1.101 × 2^3

Comparison Table

Feature	Fixed Point	Floating Point
Range	Small	Very large
Speed	Fast	Slower
Hardware	Simple	Complex
Use	Simple systems	Scientific computing
Precision	Low	High

9. BCD & BCD Addition (5 marks)

BCD (Binary Coded Decimal)

- Each decimal digit is represented by **4-bit binary**.

Example:
79 → 0111 1001

BCD Addition Steps

1. Add binary numbers normally.
2. If result > 9 or carry = 1 → add **6 (0110)**.
3. Adjust carry to next digit.

Example

```
  1001 (9)
+ 0101 (5)
--------
  1110 (14 > 9)
+ 0110
--------
1 0100  → 14 = 1 carry, 4
```

10. Decimal Arithmetic Unit (5 marks)

Decimal Arithmetic Unit (DAU) performs decimal operations inside CPU.

Functions of DAU

1. **Addition and subtraction** of decimal digits using BCD.
2. **Multiplication and division** in decimal format.

3. **Conversion** between binary ↔ decimal.
4. **Error checking** during decimal calculations.
5. Used in:
    - Financial calculations
    - Currency processing
    - Business applications

Why Needed?

Many applications require exact decimal values (money, measurement) where binary floating-point introduces rounding errors.

Ch. 5 THE MEMORY SYSTEM

1. Define RAM.

RAM (Random Access Memory) is a type of **volatile** primary memory used by the CPU to store data and instructions temporarily during execution.
It allows **read and write** operations and provides very fast access time.

Key Features

- Volatile (data lost when power is off)
- High-speed
- Directly accessed by the CPU
- Used for running programs and processes

Example: DDR4 RAM in computers.

2. What is the function of ROM in a computer system?

ROM (Read Only Memory) is **non-volatile** memory that permanently stores important instructions required for system startup.

Functions

- Stores BIOS/firmware
- Provides instructions to initialize hardware
- Keeps permanent programs that must not change
- Helps the computer boot even when power is turned off

3. List the types of semiconductor memory.

Primary classification

1. **RAM (Volatile)**
    - DRAM (Dynamic RAM)
    - SRAM (Static RAM)
2. **ROM (Non-volatile)**
    - PROM
    - EPROM
    - EEPROM
    - Flash Memory

These memories are made from semiconductor chips like silicon.

4. What does cache memory do in a computer system?

Cache memory is a small, high-speed memory located between CPU and main memory.

Functions

- Stores frequently accessed instructions/data
- Reduces CPU waiting time
- Makes program execution faster
- Uses locality of reference (temporal & spatial locality)

Cache allows the CPU to access data in nanoseconds instead of microseconds.

5. What is Virtual Memory? ***(5 marks)***

Virtual memory is a technique that allows the computer to use a portion of **secondary storage (like hard disk/SSD)** as an extension of RAM.

Key features

- Automatically expands usable memory
- Allows large programs to run on small RAM
- Uses paging and swapping techniques
- Provides illusion of large memory

6. Difference Between RAM and ROM with Examples.

Feature	RAM	ROM
Nature	Volatile	Non-volatile
Access	Read & Write	Read-only
Function	Stores running programs & data	Stores firmware/BIOS
Speed	Faster	Slower
Example	DDR4 RAM	BIOS chip, EEPROM

7. How Cache Memory Improves Performance?

Cache memory improves performance through:

(1) Reduced Memory Access Time

Cache is much faster than RAM → CPU gets data quickly.

(2) Locality of Reference

- **Temporal locality:** Recently used data is reused
- **Spatial locality:** Nearby data is likely to be used

(3) Reduced CPU Idle Time

CPU doesn’t wait for slow RAM.

(4) Multi-level Cache (L1, L2, L3)

- L1: inside CPU, fastest
- L2, L3: larger but slightly slower

Thus, cache acts as a buffer between CPU and RAM, boosting performance significantly.

8. Role of Virtual Memory in Handling Large Programs.

Virtual memory allows computers to run programs larger than the actual RAM.

How it Works

1. Splits memory into pages
2. Stores active pages in RAM
3. Keeps inactive pages on hard disk (swap space)

4. Automatically swaps data between RAM & disk
5. Gives user the illusion of “large memory”

Advantages

- Run large applications
- More multitasking capability
- Better memory management

9. How Secondary Storage Supports the Main Memory?

Secondary storage supports main memory in the following ways:

(1) Permanent Storage

Main memory is volatile, so secondary storage keeps data permanently.

(2) Provides Backup

Data stored in HDD/SSD remains safe even if power is off.

(3) Virtual Memory Support

Hard disk/SSD acts as extended RAM using swapping/paging.

(4) Large Capacity

Provides terabytes of storage, unlike limited RAM.

(5) Program Loading

Programs are first stored in secondary memory, then loaded into RAM for execution.

10. What is RAID? Describe its purpose.

RAID → **Redundant Array of Independent Disks**
A technology that combines multiple disks to improve performance and data reliability.

Purpose of RAID

1. **Data Redundancy** (protects against disk failure)
2. **Improved Read/Write Speed**
3. **High Storage Capacity**
4. **Fault Tolerance**

RAID Levels

- **RAID 0** – Striping (fast, no safety)
- **RAID 1** – Mirroring (safest)
- **RAID 5** – Parity-based (good balance)
- **RAID 10** – Mirroring + striping (best performance + safety)

11. 8GB RAM and 1TB Hard Disk — How Virtual Memory Helps When App Needs 10GB?

Assume an application needs **10GB RAM**, but system has only **8GB**.

How Virtual Memory Handles It

1. OS uses **2GB of hard disk** as temporary RAM (swap space).
2. Active pages (data currently used) stay in RAM.
3. Inactive pages are stored on disk.
4. OS swaps pages between RAM ↔ disk as needed.
5. Program runs normally even though physical RAM is less.

Result:

- Application executes without crashing
- System uses paging to manage memory
- User feels like total memory = 10GB

12. Classify as Primary or Secondary Storage.

Device	Type
SSD	Secondary Storage
DRAM	Primary Storage (RAM)
ROM	Primary Storage
HDD	Secondary Storage
Cache	Primary Storage (inside CPU)