

Structured, Semi-structured and Unstructured Data

Structures Data



Structured data is data that has a standardized format for efficient access by software and humans alike. It is typically tabular with rows and columns that clearly define data attributes. Computers can effectively process structured data for insights due to its quantitative nature.

Examples



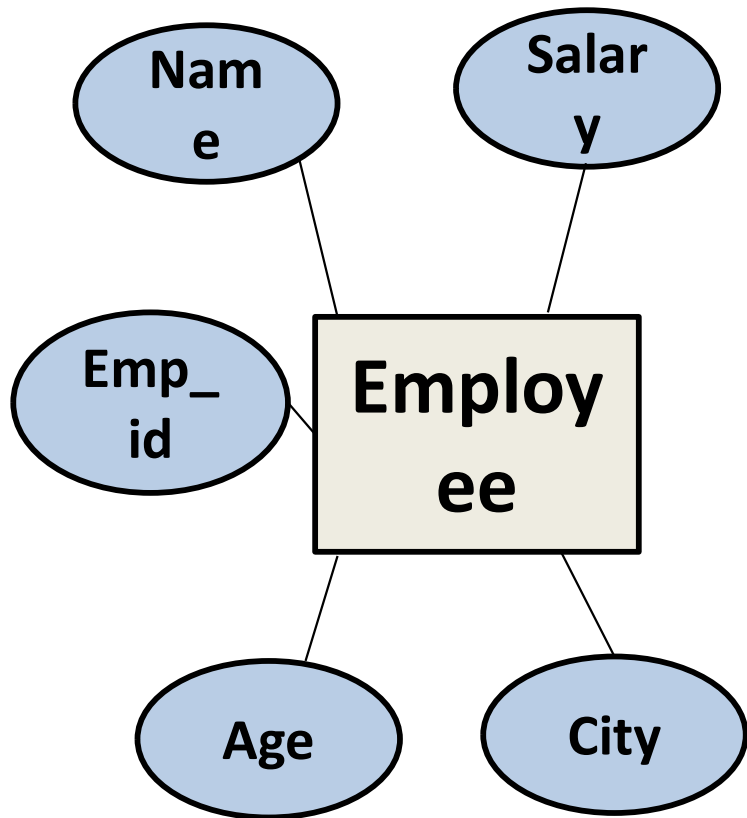
Here are examples of structured data systems:

- 1) Excel files
- 2) SQL databases
- 3) Point-of-sale data
- 4) Web form results
- 5) Search engine optimization (SEO) tags

Relational Database

- A Relational Database is a collection of data items which are organized in the form of tables of information, which can be easily accessed.
- This concept was introduced by E.F.Codd a researcher at IBM in 1970 .
- In Relational Databases the data is stored using rows and columns in the form of a table.

Name	Age	City	Salary
Santosh	29	Mumbai	34500
Rupali	30	Pune	30000
Kalpana	45	Delhi	50000
Monali	28	Surat	35000
Hemant	41	Mumbai	55000



**Employee Table:
Relation**

Columns:

Attributes

Emp_id	Name	Salary	Age	City
101	Kavita	18800	21	Mumbai
102	Piyush	21000	23	Pune

Rows:
Entities

Relation, Entities and Attributes.

1

Relation : Table
which contains rows
and columns of
related data.

2

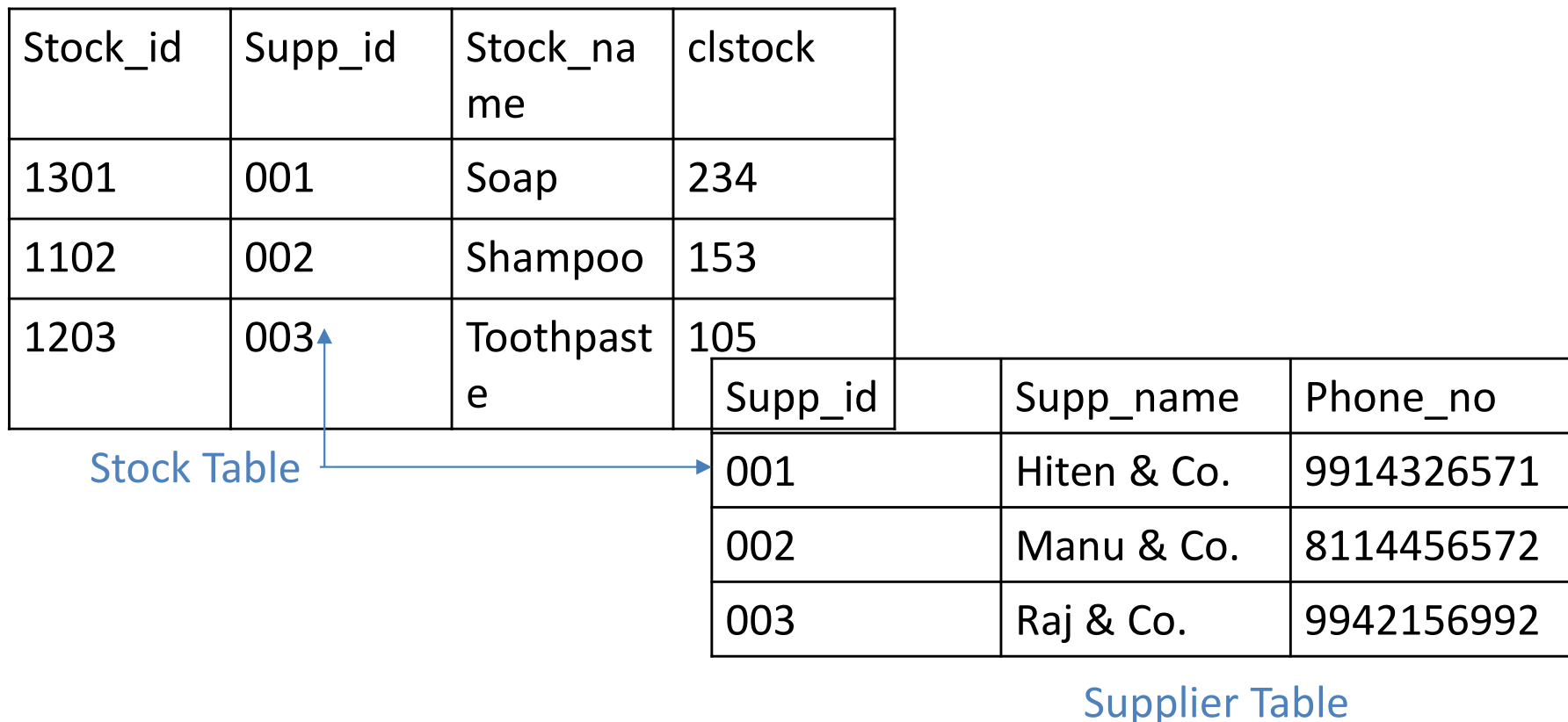
Entities (Rows) : They
are items about
which some relevant
information is stored
in the database.

3

Attributes (Columns)
: They are the
qualities of an entity
that are stored as
information.

Relationship

- A relationship is a connection between the data stored in one relational database table and another.



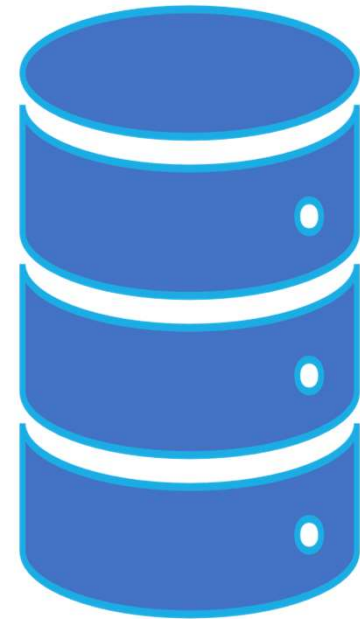
Join

- A join is a connection between two tables where the data from them is merged together based on a field (column) that is common to these tables, creating a new virtual table.
- To create a join it is necessary that the tables have a relationship.

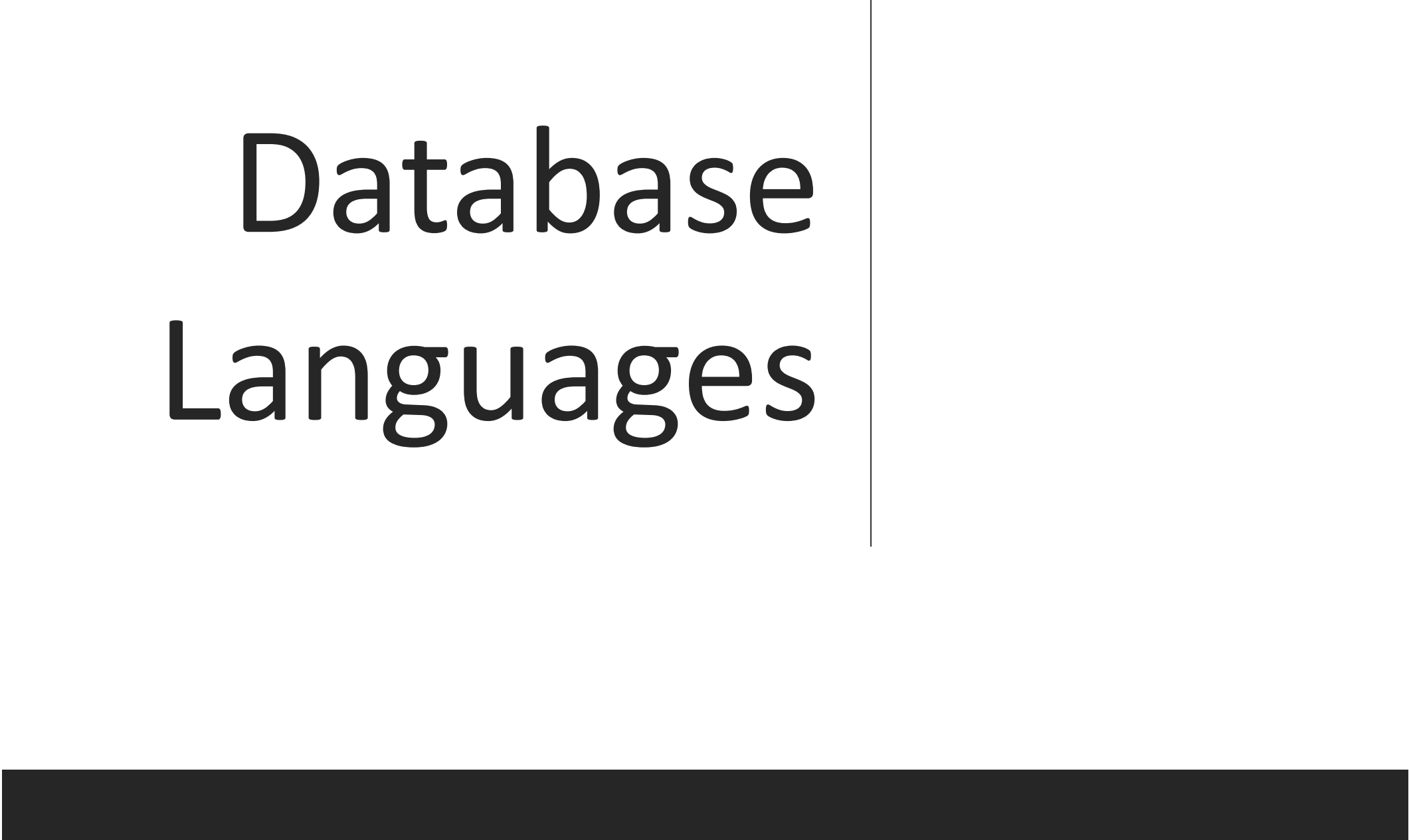
Stock_id	Supp_id	Stock_name	Supp_name	Phone_no.
1301	001	Soap	Hiten & Co.	9914326571
1102	002	Shampoo	Manu & Co.	8114456572
1203	003	Toothpaste	Raj & Co.	9942156992

Relational Database Management System.

- The Software that is used with Relational database is called a Relational Database Management System (RDBMS).
- Examples: MY SQL, MS-ACCESS etc.



Database Languages

A thin vertical line is positioned to the right of the text. At the bottom of the slide, there is a solid dark horizontal bar.

Data Definition Language (DDL)

- This language is used by the designer and programmers of the database to indicate the content and structure of the database.

Data Manipulation Language (DML)

- This language is used primarily for data manipulation and processing. It involves retrieving the data, arranging the data deleting the data and displaying the data etc.

Data Control Language (DCL)

- This is used for controlling the data and access to the database. It is used to address security issues and restrict the access to the database.

Transaction Control Language (TCL)

- TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Summary

- RDBMS applications store data in a tabular form.
- **Attributes** describe the characteristics or properties of an **entity** in a database table (**Relation**).
- A **JOIN** is used to combine rows from two or more tables, based on a related column between them.
- DDL is used for specifying the database schema.
- DML is used for accessing and manipulating data in a database.
- DCL is used for granting and revoking user access on a database.
- The changes in the database that we made using DML commands are either performed or rolled back using TCL.



Challenges with Structured data

Limited usage

The predefined structure is a benefit but can also be a challenge. Structured data can only be utilized for its intended purpose. For example, booking data can give you information about booking system finances and booking popularity. But it can't reveal which marketing campaigns were more effective in bringing in more bookings without further modification. You'll have to add marketing campaign relational data to your bookings if you want the additional insights.

Inflexibility

It can be costly and resource-intensive to change the schema of structured data as circumstances change and new relationships or requirements emerge.

What is semi-structured data?

Semi-structured data is a type of digital information that does not adhere to the rigid structure of traditional databases or spreadsheets, yet it contains some level of organization and can be processed by computers.

It falls between fully structured data, which fits neatly into tables with predefined schemas, and unstructured data, which lacks a specific format or organization.

It exhibits the following key characteristics:

- 1)Flexible schema
- 2)Human-readable
- 3)Metadata
- 4)Mix of data types
- 5)Hierarchy
- 6)Partial consistency
- 7)Scalability



1. Flexible schema

Semi-structured data does not adhere to a strict, predefined schema, allowing for variations in the structure and content of each data instance.

2. Human-readable

It is often human-readable, with elements like labels and tags, making it more accessible for both machines and humans.

3. Metadata

Semi-structured data typically contains [metadata](#), such as tags, attributes, or keys, which provide context and organization to the data elements.

4. Mix of data types

This type of data can encompass a variety of data formats, including JSON, XML, HTML, and YAML, and may include text, images, or multimedia content.



5. Hierarchy

It often exhibits hierarchical relationships, enabling the representation of nested and related data elements.

6. Partial consistency

Semi-structured data allows for partial consistency, meaning that not all data instances need to have the same attributes or structure.

7. Scalability

It is well-suited for data generated from diverse sources like IoT devices, mobile apps, and web pages, making it scalable and adaptable to evolving data needs.

Here are some common examples of semi-structured data:

- @ JSON (JavaScript Object Notation)
- @ XML (eXtensible Markup Language)
- @ CSV (Comma-Separated Values)
- @ HTML (Hypertext Markup Language)
- @ Log files
- @ NoSQL databases



1. JSON (JavaScript Object Notation)

JSON is a widely used format for representing data in a hierarchical structure composed of key-value pairs. It is easy to read and write for both humans and machines. JSON is commonly used in web APIs, configuration files, and data interchange between applications.

2. XML (eXtensible Markup Language)

XML is a versatile format for encoding structured data using tags to define elements and attributes. It allows for creating custom document structures and is commonly found in web services, RSS feeds, and configuration files.

3. CSV (Comma-Separated Values)

CSV files store tabular data with values separated by commas or other delimiters. While they lack a formal schema, they are commonly used for data exchange between spreadsheets and databases, as well as in log files.

5. HTML (Hypertext Markup Language)

HTML is primarily used for structuring web pages, but it contains valuable data elements such as meta-tags, attributes, and text content. Web scraping techniques are often employed to extract data from HTML documents.

6. Log files

Log files generated by various systems contain semi-structured data, including timestamps, events, and metadata. They are essential for system monitoring, troubleshooting, and security analysis.

7. NoSQL databases

NoSQL databases, like MongoDB and Cassandra, store data in semi-structured formats, allowing flexibility in data modeling and schema design. These databases are popular for handling unstructured and rapidly changing data.

Example : JSON



```
{
  "widget": {
    "debug": "on",
    "window": {
      "title": "Sample Konfabulator Widget",
      "name": "main_window",
      "width": 500,
      "height": 500
    },
    "image": {
      "src": "Images/Sun.png",
      "name": "sun1",
      "hOffset": 250,
      "vOffset": 250,
      "alignment": "center"
    },
    "text": {
      "data": "Click Here",
      "size": 36,
      "style": "bold",
      "name": "text1",
      "hOffset": 250,
      "vOffset": 100,
      "alignment": "center",
      "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
  }
}
```

Example: XML



The same text expressed as XML:

```
<widget>
  <debug>on</debug>
  <window title="Sample Konfabulator Widget">
    <name>main_window</name>
    <width>500</width>
    <height>500</height>
  </window>
  <image src="Images/Sun.png" name="sun1">
    <hOffset>250</hOffset>
    <vOffset>250</vOffset>
    <alignment>center</alignment>
  </image>
  <text data="Click Here" size="36" style="bold">
    <name>text1</name>
    <hOffset>250</hOffset>
    <vOffset>100</vOffset>
    <alignment>center</alignment>
    <onMouseUp>
      sun1.opacity = (sun1.opacity / 100) * 90;
    </onMouseUp>
  </text>
</widget>
```


Comparison



FEATURES	STRUCTURED	SEMI STRUCTURED	UNSTRUCTURED
Format Type	Relational Database	HTML, XML, JSON	Binary, Character
Version Management	Rows, columns, tuples	Not as common – graph is possible	Whole data
Implementation	SQL	Anonymous nodes	-
Robustness	Robust	Limited robustness	-
Storage Requirement	Less	Significant	Large
Applications	DBMS, RDF, ERP system, Data Warehouse, Apache Parquet, Financial Data, Relational Table	Server Logs, Sensor Output	No SQL, Video, Audio, Social Media, Online Forums, MRI, Ultrasound



Disadvantages of Semi Structured data

Difficult Analysis

While semi-structured data is more usable than unstructured data, it's less useful than structured data and it can be difficult to tag and index.

Limited Tooling

Artificial intelligence and machine learning (AI/ML) tools have not yet had a significant impact on the semi-structured data space—as such, existing tool stacks and models offer limited capability to transform datasets into actionable insights.

Data Security

It can be easy to overlook sensitive information contained in less visible or hidden parts of semi-structured data, and complacency or mistakes can make organizations vulnerable to security risks.

Unstructured Data



In its simplest form, refers to any data that does not have predefined structure or organization. Unlike structured data, which is organized into neat rows and columns within a database, unstructured data is an unsorted and vast information collection. It can come in different forms, such as text documents, emails, images, videos, social media posts, sensor data, etc.

Examples of Unstructured Data

Examples of unstructured data are:

- 1)Business Documents.
- 2)Emails.
- 3)Social Media.
- 4)Customer Feedback.
- 5)Webpages.
- 6)Open Ended Survey Responses.
- 7)Images, Audio, and Video.

Unstructured Data types



Unstructured data can be broadly classified into two categories:

human-generated unstructured data which includes various forms of content people create, such as text documents, emails, social media posts, images, and videos; and

machine-generated unstructured data, on the other hand, which is generated by devices and sensors, including log files, GPS data, Internet of Things (IoT) output, and other telemetry information.



Tools Used To Manage Unstructured Data

Technology category	Technology examples	Important features
APIs	Twitter API	Allows developers to access and collect public tweets, user profiles, and other data from the Twitter platform.
	Instagram API	Enables the extraction of data, such as user profiles, images, and comments
Data ingestion tools	Apache Nifi	Automates the movement and transformation of data between systems; provides a web-based interface to design, control, and monitor data flows
	Logstash	Server-side data processing pipeline; ingests data from multiple sources; sends data to various output destinations or file storage in real-time
Data lakes and NoSQL databases	Amazon S3	Scalable; low-latency access; easy integration with AWS services; virtually unlimited storage
	Google Cloud Storage	Multiple storage classes; automatic scaling; global edge-caching; easy-to-use API for data access
	MongoDB	JSON-like format; horizontal scalability; rich query language

Locality of reference

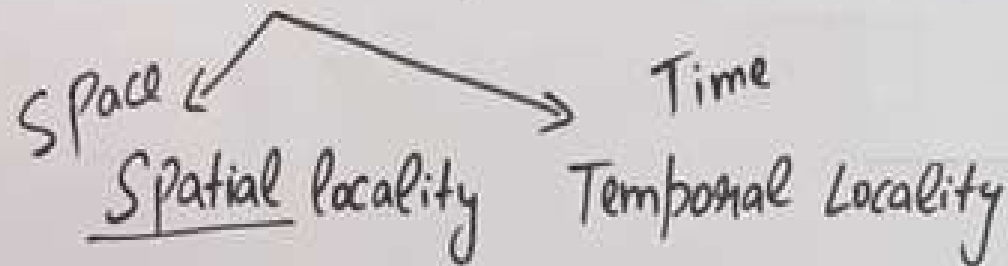


It refers to a phenomenon in which a computer program tends to access same set of memory locations for a particular time period. In other words, **Locality of Reference** refers to the tendency of the computer program to access instructions whose addresses are near one another. The property of locality of reference is mainly shown by loops and subroutine calls in a program.

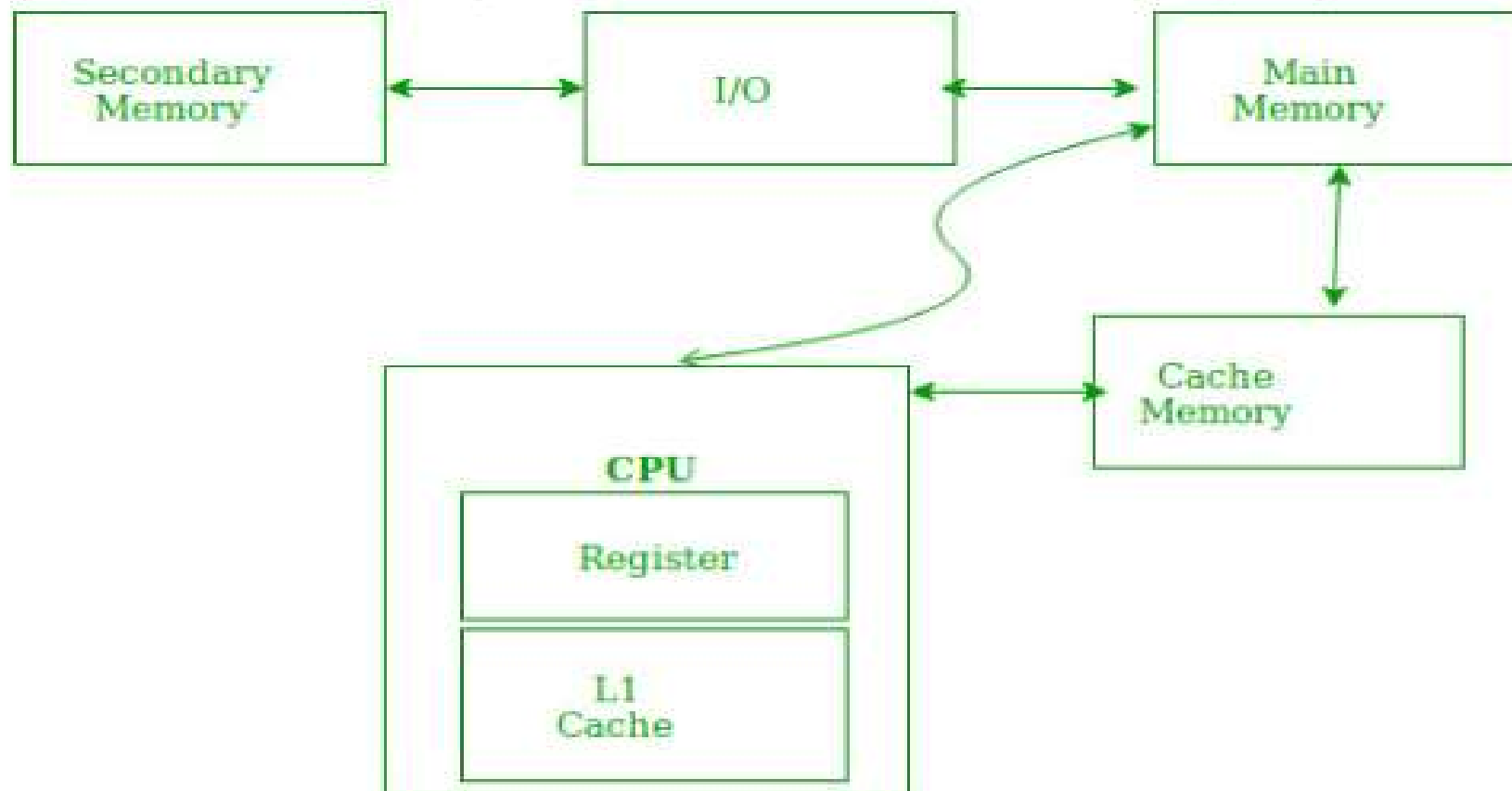
Concept



Locality of Reference

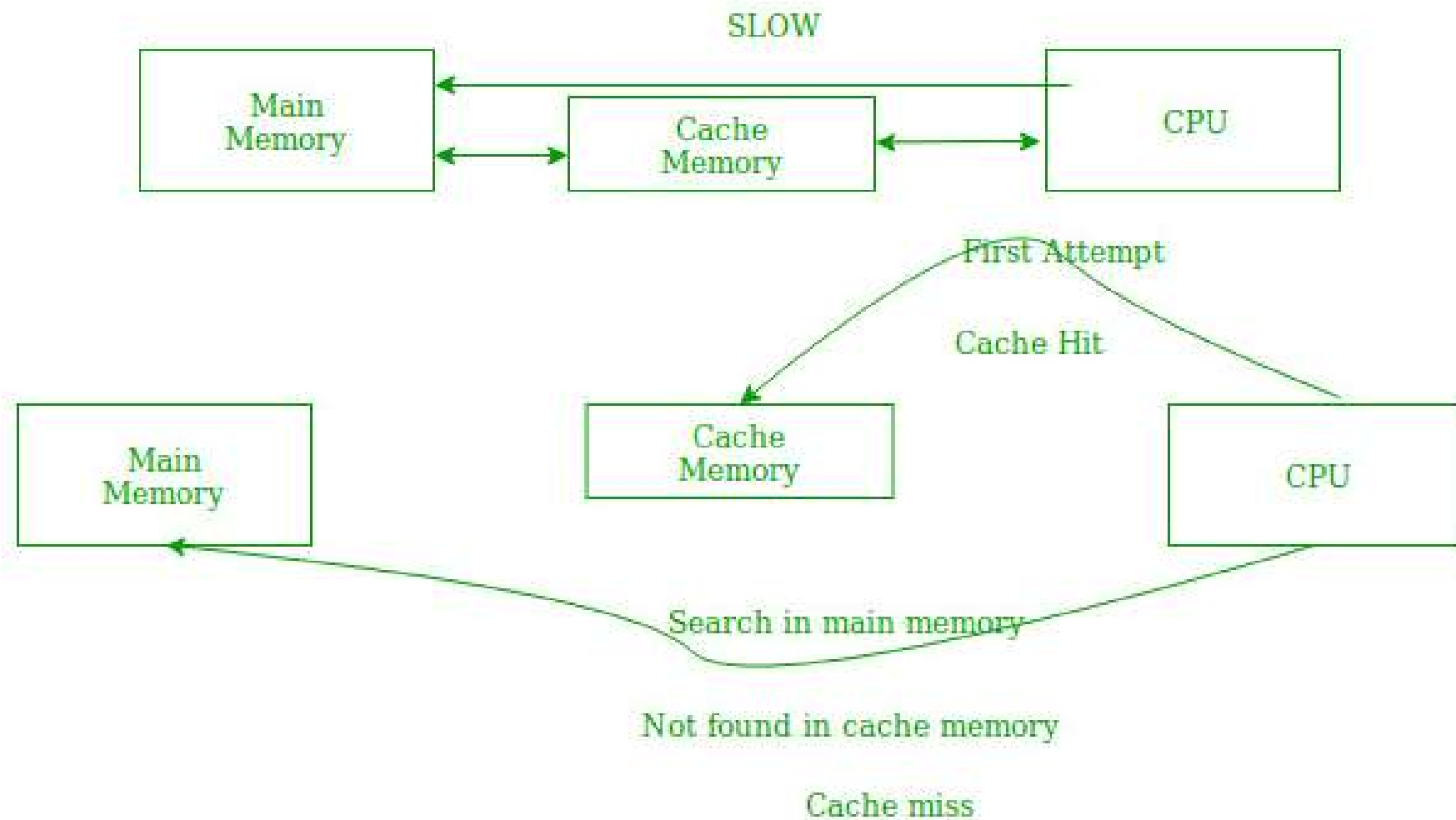


- if a word is accessed now then the word adjacent to it (close proximity) will be accessed next.
- Keeping more words in a block affects spatial locality (block size)
- if a word is referenced now then the same word will be referenced again in future
- LRU is used in temporal locality.



-
- ❑ In case of loops in program control processing unit repeatedly refers to the set of instructions that constitute the loop.
 - ❑ In case of subroutine calls, everytime the set of instructions are fetched from memory.
 - ❑ References to data items also get localized that means same data item is referenced again and again.

Flow



Cache Hit & Miss



First, it will access the cache memory as it is near to it and provides very fast access. If the required data or instruction is found, it will be fetched. This situation is known as a cache hit.

if the required data or instruction is not found in the cache memory then this situation is known as a cache miss. Now the main memory will be searched for the required data or instruction that was being searched and if found will go through one of the two ways:

First way is that the CPU should fetch the required data or instruction and use it and that's it but what, when the same data or instruction is required again. CPU again has to access the same main memory location for it and we already know that main memory is the slowest to access.

The second way is to store the data or instruction in the cache memory so that if it is needed soon again in the near future it could be fetched in a much faster way.

Cache Operation

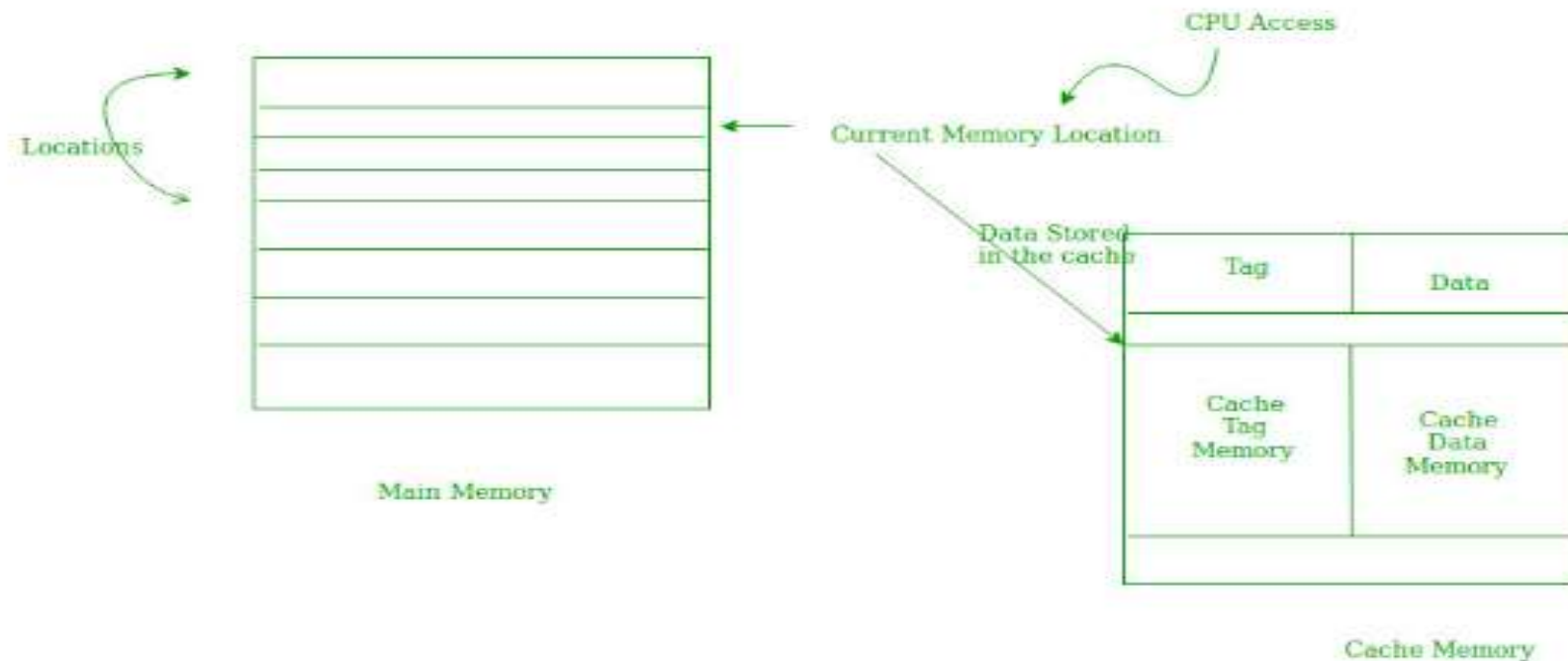


Cache Operation: It is based on the principle of locality of reference. There are two ways with which data or instruction is fetched from main memory and get stored in cache memory. These two ways are the following:

Temporal Locality



Temporal Locality – Temporal locality means current data or instruction that is being fetched may be needed soon. So we should store that data or instruction in the cache memory so that we can avoid again searching in main memory for the same data.

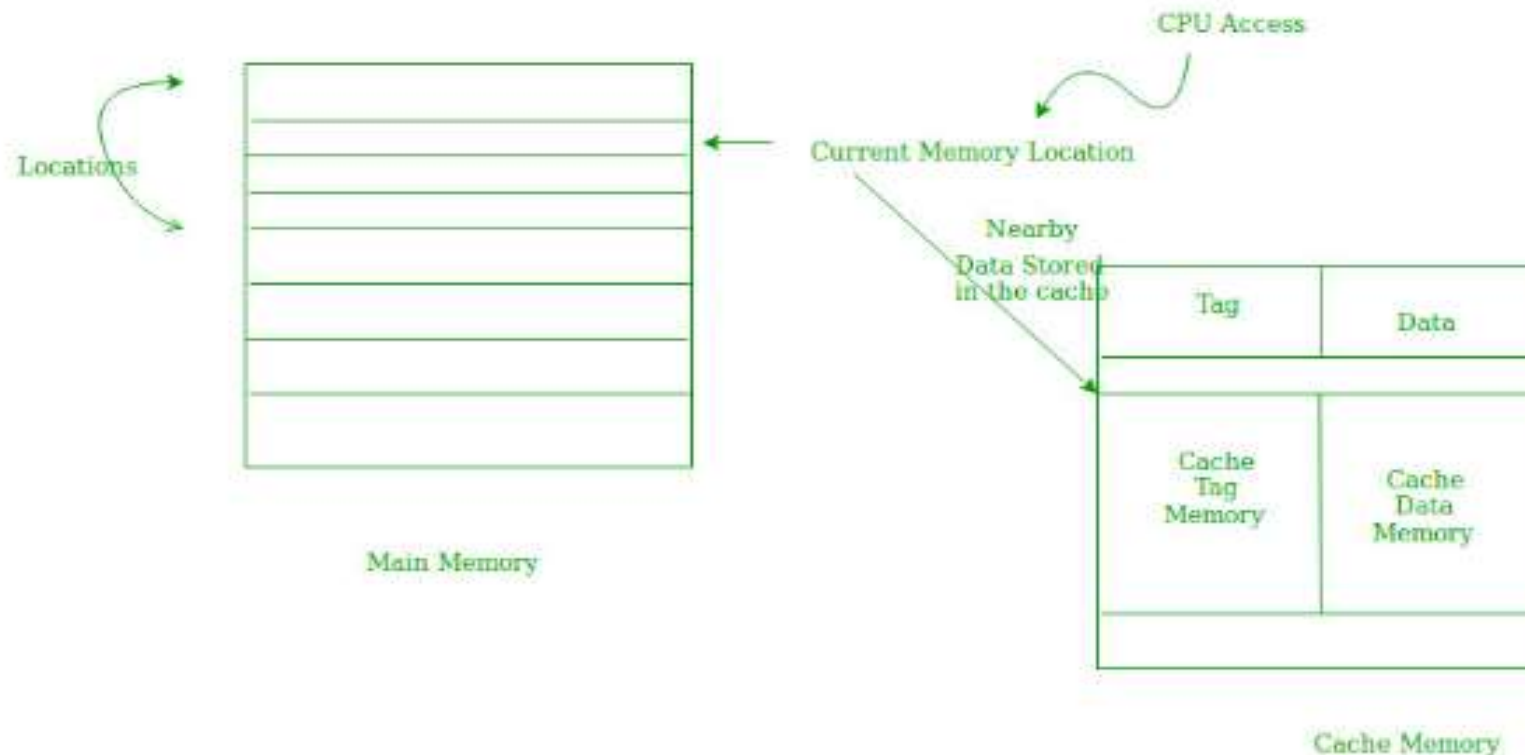


When CPU accesses the current main memory location for reading required data or instruction, it also gets stored in the cache memory which is based on the fact that same data or instruction may be needed in near future. This is known as temporal locality. If some data is referenced, then there is a high probability that it will be

Spatial Locality



Spatial Locality – Spatial locality means instruction or data near to the current memory location that is being fetched, may be needed soon in the near future. This is slightly different from the temporal locality. Here we are talking about nearly located memory locations while in temporal locality we were talking about the actual memory location that was being fetched.



Few Calculations



Hit + Miss = Total CPU Reference

Hit Ratio(h) = Hit / (Hit+Miss)

Miss Ratio = 1 - Hit Ratio(h)

Miss Ratio = Miss / (Hit+Miss)

Tavg = Average time to access memory

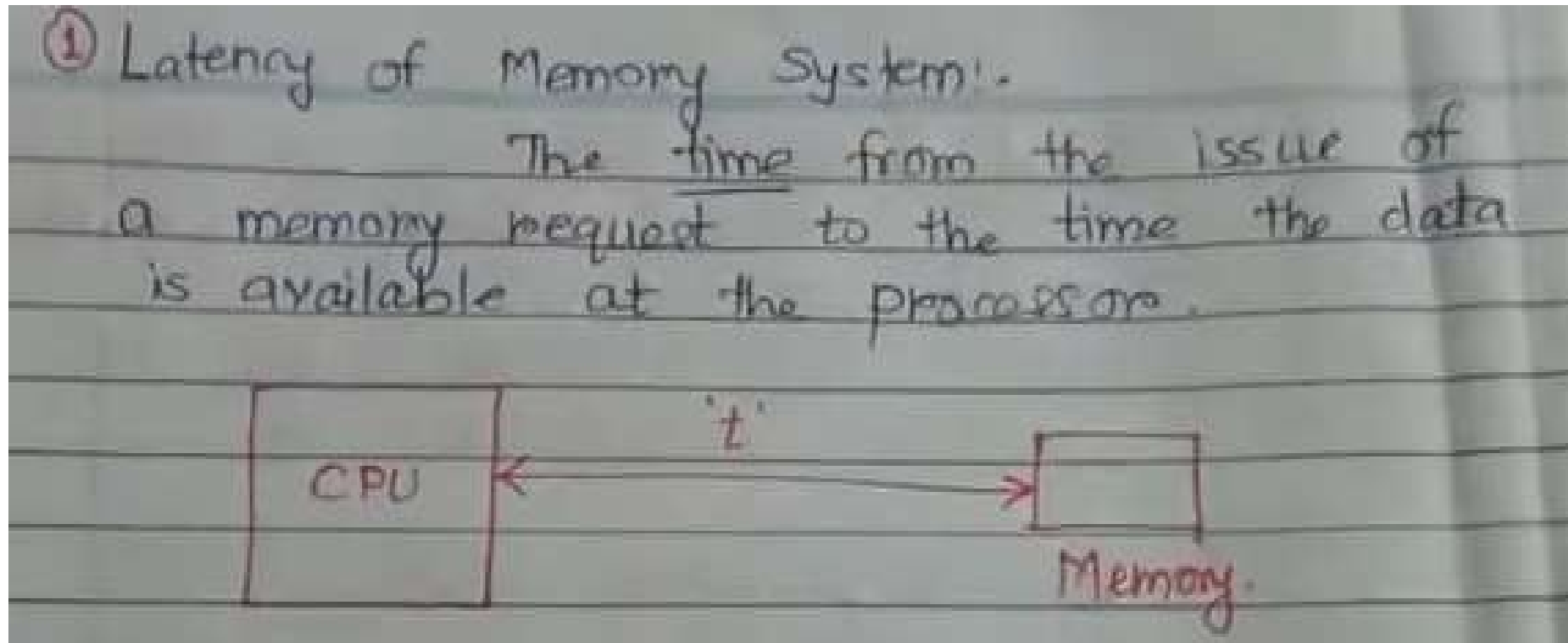
For simultaneous access

$T_{avg} = h * T_c + (1-h)*T_m$

For hierarchial access

$T_{avg} = h * T_c + (1-h)*(T_m + T_c)$

Latency Impact Example Without Cache



Contd..



② CPU clock cycle time :-
Find :- Length of 1 cycle of 1 GHz processor?
 \therefore Clock cycle time = $\frac{1}{\text{Clock Rate}}$
 $= \frac{1}{1 \times 10^9 \text{ (Hz)}}$
 $= 1 \times 10^{-9} \text{ seconds}$
 $= 1 \text{ ns.}$
 \therefore 1 GHz processor's cycle time is 1 ns.

Contd..



Effect of Memory Latency on CPU performance!

Consider,

CPU :- $1 \text{ GHz} = 1 \text{ ns}$

DRAM - 100 ns Latency. (No cache).

Processor has 2 multiply-add units.

Processor executes 4 instructions in 1 CPU cycle of 1 ns .

Soln :- 4 instructions in 1 CPU cycle

\therefore Peak Rating = 4 GFLOPS

GFLOPS : Giga Floating Operations Per Second

$$\therefore \frac{1}{100 \times 10^{-9}} = 10^7 = 10 \times 10^6$$
$$= \boxed{10 \text{ MFLOPS.}}$$

↓
CPU

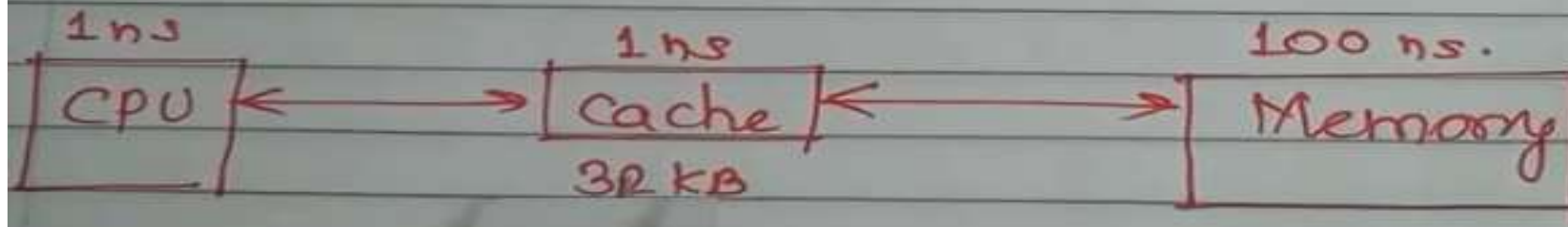
↓
CPU + Memory.

Latency Impact Example With Cache



Limitations of memory latency
Speed of 10 MFLOPS compared to
peak processor rating 4 GFLOPS.

So, cache memory is used.
Low-Latency, small and fast memory.



Consider,

Multiply two matrices (A and B)
of size 32×32 .

$$32 \times 32 = 1024 \text{ words.}$$

∴ Two matrices $1024 + 1024 = 2048 \text{ words}$
= 2 K words.

Latency Impact Example



i) Fetching two matrices into cache,

$$\begin{aligned} & 2048 \text{ words} \times 100 \text{ ns} \\ &= 204800 \times 10^{-9} \text{ sec.} \\ &= 204.8 \times 10^{-6} \text{ sec} \\ &= 204.8 \text{ } \mu\text{s} \approx \boxed{200} \text{ } \mu\text{s} \end{aligned}$$

ii) Multiplying two matrices takes $2(n)^3$ operations.

$$\begin{aligned} \therefore 2(32)^3 &= 65,535 \text{ operations,} \\ &= 64 \text{ K.} \end{aligned}$$

64000 operations and 4 operations
in 1 CPU cycle of

64000 operations and 4 operations
in 1 CPU cycle of
1 ns.

$$\frac{64000}{4} = 16000 \text{ ns} = \boxed{16 \text{ } \mu\text{s}}.$$

Contd..



iii) Total time = loading numbers + Computation (Multiplication)

$$= 200 \mu s + 16 \mu s$$
$$= 216 \mu s.$$

\therefore Peak Computation rate :-

64 K operations per 216 μs .

$$\therefore \frac{64 \text{ K}}{216 \mu s}$$
$$= \frac{64 \times 1024}{216 \times 10^{-6}}$$
$$= \frac{65536}{216} \times 10^6$$
$$= 303.40 \text{ MFLOPS}$$
$$\approx 303 \text{ MFLOPS.}$$

Conclusion

