Learning Objectives (Key Points) **1**
A1.1.6 The Process of Pipelining in Multi-Core Architectures (HL) **3**
Key Points 3
Detailed Explanation 3
1. Non-pipelined execution 3
2. Pipelined execution 4
3. Five stages of a basic instruction pipeline 4
4. Hazards in pipelining 4
5. Independent pipelines in multi-core processors 5
6. Parallel execution in multi-core processors 5
7. Why pipelining + multi-core = efficiency 6
Practice Questions 6
A1.1.7 Internal and External Types of Secondary Memory Storage **7**
Key Points 7
Detailed Explanation 7
Practice Questions 10
A1.1.8 Compression **12**
Key Points 12
Detailed Explanation 12
Summary of Methods 15
Practice Questions 16
A1.1.9 Types of Services in Cloud Implementation **16**
Key Points 16
Detailed Explanation 16

Learning Objectives (Key Points)

A1.1.6 Secondary Storage

- Identify different types of secondary storage (HDD, SSD, optical, cloud).
- Explain their characteristics: speed, capacity, cost, durability.
- Compare use cases for each type of storage.

A1.1.7 Internal and External Types of Secondary Memory Storage

- Distinguish between internal and external storage.
- Describe technologies: HDDs, SSDs, M.2, eMMC, external drives, optical discs, memory cards, NAS.
- Evaluate advantages and disadvantages of each in different contexts.
  A1.1.8 Compression
- Define compression and explain its purpose.
- Distinguish between lossless and lossy compression.
- Describe methods: RLE, Huffman coding, arithmetic coding, dictionary-based (lossless); transform coding, perceptual coding, fractal compression (lossy).
- Evaluate advantages and disadvantages of compression for different file types.

A1.1.9 Types of Services in Cloud Implementation

- Define cloud computing and its benefits.
  Describe SaaS, PaaS, and IaaS.
- Distinguish between the levels of control, flexibility, and responsibility in each model.
- Evaluate the suitability of different cloud services for various users and organizations.

A1.1.6 The Process of Pipelining in Multi-Core Architectures (HL)

Key Points

- Pipelining increases CPU throughput by overlapping instruction stages.
- Analogy: Assembly line or carwash – different tasks performed simultaneously on different items.
- Stages of a pipeline: Fetch, Decode, Execute, Memory Access, Write-back.
- Pipelining is limited by hazards (structural, data, control).
- In multi-core processors, each core runs its own pipeline.
- Parallel execution: multiple cores cooperate on one large task or process multiple tasks at once.

Detailed Explanation

1. Non-pipelined execution

- A **non-pipelined CPU** processes one instruction at a time, sequentially.
- Example:
    - Instruction 1: Fetch → Decode → Execute → Memory → Write-back.
    - Only after it fully finishes does Instruction 2 begin.
- Drawback: resources are idle. For example, while the ALU executes, the fetch unit is doing nothing.
- Analogy: In a carwash, the washer must wait for the cleaner to finish before washing the next car.
- Result: throughput is low — time to complete instructions = sum of all stage times.

2. Pipelined execution

- In **pipelining**, each stage works on a different instruction at the same time.
- Once Stage 1 finishes fetching Instruction A, it immediately fetches Instruction B, while Stage 2 decodes Instruction A.
- Instructions overlap in execution, like cars moving through an assembly line.
- Formula for throughput:
    - Non-pipelined: 1 instruction per *n* cycles.
    - Pipelined: ~1 instruction per cycle (ideal case).
- Example: If each stage takes 5 cycles, non-pipelined execution of 5 instructions = 25 cycles.
  With pipelining: ≈ 9 cycles.

3. Five stages of a basic instruction pipeline

1. **Fetch** – Program Counter (PC) provides the next instruction's address, retrieved from memory.
2. **Decode** – Instruction is broken down into opcode and operands; Control Unit generates necessary signals.
3. **Execute** – Arithmetic/Logic Unit (ALU) performs required operations.
4. **Memory access** – If the instruction involves data (load/store), memory is accessed.
5. **Write-back** – Result is written back to a register for future use.

- These stages are separated by **pipeline registers (buffers/latches)**, which hold intermediate results.
- **Clock synchronization** ensures all stages advance together.

4. Hazards in pipelining

While pipelining increases throughput, it introduces challenges:

- **Structural hazards**: when hardware resources are shared (e.g., only one memory unit for both instruction fetch and data access).
- **Data hazards**: when an instruction depends on the result of a previous one still in the pipeline.
- **Control hazards**: caused by branch/jump instructions — CPU may fetch the wrong instruction.

Solutions include **pipeline stalls (bubbles)**, **branch prediction**, and **out-of-order execution**.

5. Independent pipelines in multi-core processors

- Each core in a multi-core CPU has its **own independent pipeline**.
- This means one core can be decoding an instruction while another is executing a different one.
- Analogy: multiple carwash bays, each working independently.
- Benefit: better utilization of CPU resources — one heavy task doesn't block the entire CPU.

6. Parallel execution in multi-core processors

- Beyond independence, cores can **work together** on a large task.
- Example:
    - Core 1 processes part of a dataset, Core 2 another part, Core 3 yet another.
    - Threads are distributed across cores (multi-threading).
- Analogy: A large car being serviced by multiple teams simultaneously — one team washes, another cleans the interior, another does the wheels.
- Result: Complex workloads like video rendering, AI computations, and scientific modeling complete much faster.

7. Why pipelining + multi-core = efficiency

- Pipelining increases efficiency **within each core**.
- Multi-core architectures increase efficiency **across the whole CPU**.
- Combined, they enable:
    - Faster execution without needing much higher clock speeds.
    - Energy efficiency (parallel work at lower frequency).
    - Scalability — more cores = more throughput.

Practice Questions

1. Define pipelining and explain why it improves CPU performance compared to a non-pipelined design.
2. Using the carwash analogy, describe the difference between pipelined and non-pipelined instruction execution.
3. Outline the five stages of the instruction pipeline and explain the role of pipeline registers.
4. Explain the main types of pipeline hazards and how they can affect CPU performance.
5. Describe how independent pipelines in multi-core processors enhance computational efficiency.
6. Explain with an example how parallel execution across multiple cores can reduce the time required for complex tasks.
7. Evaluate the combined effect of pipelining and multi-core architectures on modern CPU performance.

A1.1.7 Internal and External Types of Secondary Memory Storage

Key Points

- Secondary memory = non-volatile, long-term storage.
- Two categories: **internal** (inside system) and **external** (portable/network-based).
- **Internal examples**: HDDs, SSDs, M.2 NVMe SSDs, eMMC.
- **External examples**: HDDs, SSDs, optical discs, memory cards, NAS.
- Each storage type differs in **technology, speed, cost, durability, portability, and capacity**.

Detailed Explanation

1. Nature of Secondary Storage
Secondary storage is **non-volatile**, meaning data is retained when the system is powered off. This distinguishes it from **primary memory (RAM, cache)**, which is volatile and only stores data temporarily.
Functions include:

- Storing operating systems permanently.
- Holding application software.
- Saving user data (documents, media, backups).
- Enabling archival and retrieval of information.

It exists as **internal** storage (built into the device) or **external** storage (removable or network-based).

2. Internal Storage

Hard Disk Drives (HDDs)

- Technology: spinning magnetic platters, with a read/write head that moves mechanically.

- Advantages: very high storage capacity (up to several terabytes), lower cost per gigabyte.
- Disadvantages: slow read/write speeds (50–150 MB/s), mechanical failure risk, heavy, generates noise and heat.
- Use cases: bulk data storage like movies, backups, archives.

Solid State Drives (SSDs)

- Technology: NAND flash memory (no moving parts).
- Advantages: fast read/write speeds (200–500 MB/s for SATA SSDs, much higher for NVMe), silent, durable, energy-efficient.
- Disadvantages: more expensive per gigabyte, generally smaller storage capacities.
- Use cases: operating systems, software, games requiring high-speed data access.

M.2 SSDs and NVMe

- Form factor: small and thin (looks like a stick of gum).
- NVMe (Non-Volatile Memory Express) connects directly to PCIe lanes on the motherboard, achieving speeds several times faster than SATA SSDs.
- Advantages: compact, extremely fast, efficient.
- Use cases: high-performance laptops, gaming PCs, workstations.

eMMC (Embedded MultiMediaCard)

- Technology: NAND flash memory soldered directly onto the motherboard.
- Advantages: cheap, compact, adequate for basic tasks.
- Disadvantages: limited capacity, slower than SSDs, non-replaceable.
- Use cases: budget smartphones, tablets, entry-level laptops.

---

3. External Storage

External HDDs and SSDs

- External HDDs: same technology as internal HDDs, but portable. High capacity, cheap, slower, fragile.
- External SSDs: portable, durable, fast, silent, more expensive.
- Use cases: file transfers, portable backups, expanding storage capacity.

Optical Discs (CD, DVD, Blu-ray)

- Technology: data stored in pits on reflective discs, read by lasers.
- Advantages: cheap per disc, useful for archiving, easy distribution.
- Disadvantages: slow, fragile (scratches damage data), declining relevance since fewer modern devices include optical drives.
- Use cases: media playback, long-term archiving.

Memory Cards (SD, microSD, CompactFlash)

- Technology: NAND flash memory, portable form factor.
- Advantages: compact, durable (resistant to shock, water, temperature extremes), reusable.
- Disadvantages: slower than SSDs, limited capacity compared to HDDs.
- Use cases: cameras, smartphones, portable devices.

Network Attached Storage (NAS)

- Technology: dedicated storage device connected to a network, often with multiple HDDs or SSDs.
- Features: configured with RAID (redundancy + performance), accessible by multiple users simultaneously.
- Advantages: centralized storage, expandable, suitable for collaboration and backups.

- Disadvantages: more expensive setup, requires networking knowledge.
- Use cases: businesses, households with multiple users, backup servers, media streaming hubs.

---

4. Comparison Summary

- **HDD vs SSD**: HDD cheaper and higher capacity, SSD faster, more reliable.
- **M.2 NVMe vs SATA SSD**: NVMe much faster, compact, but more costly.
- **eMMC vs SSD**: eMMC soldered, cheaper, slower, non-replaceable; SSD modular, faster, higher quality.
- **Memory cards vs Optical discs**: memory cards durable, portable, rewritable; optical discs fragile, slower, cheaper for archiving.
- **External drives vs NAS**: external drives for individuals; NAS for multi-user environments needing centralized, scalable solutions.

Practice Questions

1. **Describe** how data is stored and accessed on a hard disk drive (HDD).
2. **Explain** the advantages of solid state drives (SSDs) over hard disk drives (HDDs).
3. **Distinguish** between SATA SSDs and M.2 NVMe SSDs in terms of performance and installation.
4. **Outline** the role of eMMC storage in budget devices such as smartphones and entry-level laptops.
5. **Compare** the advantages and disadvantages of memory cards and optical discs for portable storage.
6. **Explain** how a Network Attached Storage (NAS) system operates in a business environment.

7. **Evaluate** which storage option (HDD, SSD, or NAS) would be most suitable for a professional photographer and justify your choice.

A1.1.8 Compression

Key Points

- Compression reduces file size by encoding data with fewer bits
- Two main types: **lossless** and **lossy**
- Benefits: saves storage, speeds up data transfer, reduces costs
- Common methods: **Run-Length Encoding (RLE)**, **Huffman Coding**, **Arithmetic Coding**, **Transform Coding** (JPEG, MP3), **Dictionary-based methods (LZW, ZIP)**

Detailed Explanation

1. Purpose of Compression
Compression is the process of encoding data using fewer bits than its original representation.

- Saves **storage space** (important for devices with limited capacity).
- Reduces **transmission time** (useful in networking and cloud applications).
- Lowers **costs** of storage and bandwidth.
  For example: compressing a 10 MB file into 2 MB allows it to be stored more easily and transmitted more quickly.

---

2. Lossless Compression
Lossless compression reduces file size **without losing any data**. The original can be reconstructed exactly.

- Works by removing redundancy and representing repeating patterns more efficiently.
- Examples of use: text files, databases, executable programs (accuracy required).

Methods of Lossless Compression:

1. **Run-Length Encoding (RLE)**

- Replaces repeated sequences with a count + character.
- Example: `AAAAABBBCCDAA` → `5A3B2C1D2A`
- Advantage: very effective for simple images, documents, and faxes.
- Limitation: not efficient for complex data with little repetition.

2. **Huffman Coding**
    - Assigns shorter codes to frequent symbols and longer codes to less frequent ones.
    - Creates a binary tree where the most common symbols have the shortest paths.
    - Used in JPEG and MP3 encoding as part of final entropy coding.
3. **Arithmetic Coding**
    - Represents an entire message as a single fractional number between 0 and 1.
    - More efficient than Huffman when symbol probabilities are not powers of 2.
    - Used in advanced compression formats (e.g., JPEG2000).
4. **Dictionary-based Compression (LZW, ZIP, GIF)**
    - Builds a dictionary of repeated sequences in the file.
    - Each sequence is replaced by a reference to the dictionary.
    - Example: The word “compression” repeated 100 times is stored once, and each later occurrence is replaced with a pointer.

---

3. Lossy Compression

Lossy compression **removes non-essential data** permanently to achieve greater reduction.

- The decompressed file is not identical to the original, but close enough for human perception.

- Examples: images, audio, video — where small losses in quality are acceptable.

Methods of Lossy Compression:

1. **Transform Coding (JPEG, MP3, MPEG)**
    - Example: JPEG image compression
    - Steps:
        1. Divide image into small blocks (e.g., 8×8 pixels).
        2. Apply Discrete Cosine Transform (DCT) to convert spatial data (pixels) into frequency components.
        3. Quantization reduces precision of high-frequency components (fine details less noticeable to the human eye).
        4. Entropy coding (zigzag scan → RLE → Huffman coding) further compresses.
    - Advantage: huge reduction in file size.
    - Disadvantage: repeated compression causes visible artifacts (blurring, blockiness).
2. **Perceptual Coding (used in audio/video)**
    - Removes data beyond human hearing/vision sensitivity.
    - MP3: eliminates inaudible frequencies.
    - MPEG/MP4: removes redundant video frames and uses motion prediction.
3. **Fractal Compression** (images)
    - Uses self-similarity patterns within images to encode them with mathematical functions.
    - Can achieve very high compression, but computationally expensive.

---

4. Advantages of Compression

- Reduces file storage requirements.
- Makes data transfer faster (important for streaming).
- Lowers cost of storage and bandwidth.
- Enables efficient archiving and backups.

5. Disadvantages of Compression

- Lossy methods reduce quality permanently.
- Compression/decompression requires processing time and resources.
- Some algorithms may increase file size if the data has little redundancy (e.g., RLE on random data).
- Compatibility issues may arise with older systems.

Summary of Methods

Lossless Methods:

- Run-Length Encoding (RLE)
- Huffman Coding
- Arithmetic Coding
- Dictionary-based compression (LZW, ZIP)

Lossy Methods:

- Transform Coding (JPEG, MP3, MPEG)
- Perceptual Coding (audio/video)
- Fractal Compression (images)

Practice Questions

1. **Define** the term compression and state its main purpose.
2. **Distinguish** between lossless and lossy compression, using examples of each.
3. **Describe** how Run-Length Encoding (RLE) works with a worked example.
4. **Explain** how Huffman coding compresses data by assigning shorter codes to frequent symbols.
5. **Outline** the role of quantization in JPEG transform coding.
6. **Compare** the effectiveness of dictionary-based compression (e.g., ZIP) and RLE for text data.
7. **Explain** how perceptual coding reduces file size in MP3 audio compression.
8. **Evaluate** the impact of lossy compression on multimedia services such as YouTube or Spotify.

A1.1.9 Types of Services in Cloud Implementation

Key Points

- Cloud computing provides IT resources and services via the internet.
- Three main models: **Software as a Service (SaaS)**, **Platform as a Service (PaaS)**, **Infrastructure as a Service (IaaS)**.
- Each offers a different balance of control, flexibility, and responsibility.

Detailed Explanation

1. What is Cloud Computing?
Cloud computing allows organizations and individuals to use computing resources (software, storage, servers, platforms) delivered over the internet instead of owning and managing physical infrastructure.

- Provides **scalability** (resources can be increased or decreased on demand).
- Offers **cost efficiency** (subscription model instead of heavy upfront investment).
- Increases **accessibility** (services available anywhere with internet access).

---

2. Software as a Service (SaaS)
SaaS delivers fully managed applications over the internet.

- Users access applications through a **web browser or mobile app**.
- No need to install, update, or maintain software locally.
- Common examples: Google Workspace (Docs, Gmail, Drive), Microsoft 365, Dropbox, Salesforce.

Advantages:

- Accessible from anywhere, on any device.
- Lower upfront costs (subscription model).
- Automatic updates and maintenance handled by the provider.
- Scalable for small to large businesses.

Disadvantages:

- Requires a reliable internet connection.
- Limited customization (compared to locally installed software).
- Data security and privacy depend on the provider.

---

3. Platform as a Service (PaaS)
PaaS provides a **development and deployment environment** in the cloud.

- Developers can build, test, and deploy applications without managing hardware.
- Includes middleware, databases, frameworks, and tools.
- Example: Microsoft Azure App Service, Google App Engine, Heroku.

Advantages:

- Developers focus on **coding**, not infrastructure.
- Speeds up development and deployment cycles.
- Easy scalability as applications grow.
- Built-in collaboration tools for teams.

Disadvantages:

- Vendor lock-in (difficult to switch providers once an app is built on one platform).
- Less control over the hosting environment.
- May not suit applications requiring specialized configurations.

---

4. Infrastructure as a Service (IaaS)
IaaS provides **virtualized computing resources** (servers, storage, networking) over the internet.

- Businesses rent virtual machines instead of buying physical servers.
- Users manage their own software, applications, and security while the provider manages the infrastructure.
- Example: Amazon Web Services (AWS) EC2, Google Cloud Compute Engine, Microsoft Azure Virtual Machines.

Advantages:

- Full control over operating systems, applications, and settings.

- Scalable to meet growing user or business needs.
- Cost-effective — avoids upfront hardware costs.
- Flexible for businesses that require custom setups.

Disadvantages:

- Requires technical expertise to configure and manage.
- Security is partly the responsibility of the user.
- More complex than SaaS or PaaS for non-technical users.

---

5. Comparison of SaaS, PaaS, and IaaS

Feature	SaaS	PaaS	IaaS
What it offers	Ready-to-use software apps	Development platform + tools	Virtualized servers, storage, networks
User responsibility	Just use the software	Build/deploy applications	Full control of OS, apps, and data
Examples	Gmail, Microsoft 365, Dropbox	Azure App Service, Google App Engine	AWS EC2, Google Cloud Compute
Best for	End-users and businesses needing apps	Developers building applications	IT teams needing infrastructure

6. Real-World Applications

- **SaaS:** Schools using Google Workspace to collaborate online.
- **PaaS:** A startup developing a mobile app using Heroku's cloud tools.
- **IaaS:** A large enterprise hosting its e-commerce website on AWS to handle millions of transactions securely.

Practice Questions

1. **Define** the term cloud computing.
2. **Describe** the features of Software as a Service (SaaS).
3. **Explain** how Platform as a Service (PaaS) supports software development.
4. **Compare** the roles of SaaS and IaaS in providing cloud-based services.
5. **Outline** one advantage and one disadvantage of using SaaS for businesses.
6. **Distinguish** between PaaS and IaaS in terms of user control and responsibility.
7. **Evaluate** the potential benefits and risks of adopting cloud services for a medium-sized company.