Department of Computer Science & Engineering

Details of Lesson Plan

S.No.	Particulars	Details
1.	Course Name	Modern Computer Architecture
2.	Course Code	COM-401
3.	Academic Year	2024-2025
4.	Semester	4
5.	Number of Lesson plans	45
6.	Faculty Assigned	Dr. Mir Aadil

Faculty Signature

Lesson Plan No. 1.1	Course Name: Modern Computer Architecture Topic: Basics of Designing Combinational and Sequential Logic	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the fundamentals of combinational and sequential logic design. b. Identify the differences between combinational and sequential circuits. c. Design basic combinational and sequential circuits.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Discuss the importance of logic circuits in computer systems.
	• Define combinational and sequential logic.
	Development (30 minutes)
	• **Combinational Logic (15 minutes):**
	  • Explain the design process of combinational circuits.
	  • Work through examples: Adders, Subtractors.
	  • Real-time example: Show a basic adder circuit using Logisim.
	• **Sequential Logic (15 minutes):**
	  • Explain flip-flops, counters, and registers.
	  • Design a simple sequential circuit using JK flip-flops.
	Exercise (5 minutes)
	• Ask students to design a 4-bit adder using combinational logic.
	• Review and discuss designs.

Closure	Summarize key points: • Summarize key points: differences between combinational and sequential logic. • **Suggested Readings:** • *Digital Design and Computer Architecture* by David Harris & Sarah Harris, Chapter 3.
Evaluation	Reflective question: Why is sequential logic critical in digital systems?

Lesson Plan No. 1.2	Course Name: Modern Computer Architecture Topic: Computer Registers and Instructions	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the purpose of computer registers. b. Learn about different types of instructions and their functions. c. Design a basic instruction execution process.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Define computer registers and their role in instruction execution.
	• Discuss different types of instructions (RISC vs. CISC).
	
	Development (30 minutes)
	• **Registers (15 minutes):**
	  • Explain different types of registers: Accumulator, Data, Address.
	  • Diagram: Show how instructions interact with registers.
	• **Instruction Types (15 minutes):**
	  • Explain arithmetic, logical, and control instructions.
	  • Real-time example: Execution of a simple arithmetic instruction.
	
	Exercise (5 minutes)
	• Ask students to design a 4-bit adder using combinational logic.
	• Review and discuss designs.

Closure	Summarize key points: • Summarize the function of registers and types of instructions. • **Suggested Readings:** • *Computer Organization and Design* by David Patterson & John Hennessy, Chapter 5.
Evaluation	Reflective question: Write the execution process of an arithmetic instruction using registers.

Lesson Plan No. 1.3	Course Name: Modern Computer Architecture Topic: Timing and Control, Instruction Cycle	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Learn about timing and control in instruction execution. b. Understand the instruction cycle and its phases..
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Define timing and control in a computer system.
	• Overview of the instruction cycle.
	Development (30 minutes)
	• **Timing and Control (15 minutes):**
	  • Explain the role of clock cycles in timing.
	  • Discuss control signals and their generation.
	• **Instruction Cycle (15 minutes):**
	  • Explain fetch, decode, execute phases.
	  • Demonstration: Simulate an instruction cycle.
	Exercise (5 minutes)
	  • Students draw a timing diagram for a simple instruction.

Closure	Summarize key points: • Summarize timing control and the instruction cycle. • **Suggested Readings:** • *Computer System Architecture* by M. Morris Mano, Chapter 4.
Evaluation	Reflective question: How does timing affect instruction execution?

Lesson Plan No. 1.4	Course Name: Modern Computer Architecture Topic: Memory Reference Instructions and I/O Interruption	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand memory reference instructions. b. Learn the concept of I/O interruption and its significance.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Define memory reference instructions and their role in a CPU.
	• Overview of I/O interruption.
	Development (30 minutes)
	• **Memory Reference Instructions (15 minutes):**
	  • Explain instructions like LDA, STA, ADD.
	  • Real-time example: Execution of a memory reference instruction.
	• **I/O Interruption (15 minutes):**
	  • Explain the types of interrupts.
	  • Diagram: Show how an interrupt is handled by the CPU.
	Exercise (5 minutes)
	Students identify and describe a scenario for an I/O interrupt.

Closure	Summarize key points: • Summarize memory reference instructions and I/O interruptions. • **Suggested Readings:** • *Computer Organization and Architecture* by William Stallings, Chapter 6.
Evaluation	Reflective question: Write a short note on the handling of I/O interrupts.

Lesson Plan No. 1.5	Course Name: Modern Computer Architecture Topic: Adder and Subtractor Circuits	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the design and function of adder and subtractor circuits. b. Implement simple adder and subtractor circuits.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Discuss the importance of logic circuits in computer systems
	• Discuss the importance of adder and subtractor circuits in digital systems.
	
	Development (30 minutes)
	• **Adder Circuits (15 minutes):**
	  • Explain half-adder and full-adder circuits.
	  • Real-time example: Design and simulate a full-adder circuit.
	• **Subtractor Circuits (15 minutes):**
	  • Explain half-subtractor and full-subtractor circuits.
	  • Design and simulate a full-subtractor circuit.
	
	Exercise (5 minutes)
	• Ask students to design a 4-bit adder using combinational logic.
	• Review and discuss designs.

Closure	Summarize key points: • Summarize key points: differences between combinational and sequential logic. • **Suggested Readings:** • *Digital Design and Computer Architecture* by David Harris & Sarah Harris, Chapter 3.
Evaluation	Reflective question: Why is sequential logic critical in digital systems?

Lesson Plan No. 1.6	Course Name: Modern Computer Architecture Topic: Booth Multiplication Algorithm	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Learn about the Booth multiplication algorithm. b. Implement the Booth algorithm for binary multiplication.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Introduce the need for efficient multiplication in processors..
	• Define combinational and sequential logic.
	Development (30 minutes)
	• **Booth Algorithm (20 minutes):**
	  • Explain the algorithm with examples.
	  • Real-time example: Walk through binary multiplication using Booth's algorithm.
	• **Application (10 minutes):**
	  • Demonstrate how the Booth algorithm is used in CPU multipliers.
	Exercise (5 minutes)
	• Students solve a multiplication problem using Booth's algorithm.

Closure	Summarize key points: • Summarize the Booth algorithm and its advantages. Suggested Readings: • Computer Arithmetic Algorithms by Israel Koren, Chapter 3.
Evaluation	Reflective question: Why is sequential logic critical in digital systems?

Lesson Plan No. 1.7	Course Name: Modern Computer Architecture Topic: Pipelining, Control Hazards	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of pipelining in CPU design. b. Identify control hazards and their mitigation techniques.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Define pipelining and its significance in CPU performance.
	
	• **Development (30 minutes)**
	• **Pipelining (10 minutes):**
	  • Explain the stages of a pipeline.
	  • Real-time example: Simulate a pipelined instruction execution.
	
	• **Control Hazards (10 minutes):**
	  • Identify control hazards and discuss mitigation techniques (e.g., branch prediction).
	
	• **Cache Basics (10 minutes):**
	  • Explain cache memory, levels, and characteristics.
	
	Exercise (5 minutes)
	
	Students diagram a simple 3-stage pipeline and identify hazards.

Closure	Summarize key points: • Summarize pipelining, control hazards, and cache basics. • **Suggested Readings:** *Computer Architecture: A Quantitative Approach* by John L. Hennessy & David A. Patterson, Chapter 3.
Evaluation	Reflective question: How does pipelining improve CPU performance?

Lesson Plan No. 1.8	Course Name: Modern Computer Architecture Topic: Cache Basics	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the fundamentals of combinational and sequential logic design. b. Identify the differences between combinational and sequential circuits. c. Design basic combinational and sequential circuits.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Discuss the importance of logic circuits in computer systems.
	• Define combinational and sequential logic.
	Development (30 minutes)
	• **Combinational Logic (15 minutes):**
	  • Explain the design process of combinational circuits.
	  • Work through examples: Adders, Subtractors.
	  • Real-time example: Show a basic adder circuit using Logisim.
	• **Sequential Logic (15 minutes):**
	  • Explain flip-flops, counters, and registers.
	  • Design a simple sequential circuit using JK flip-flops.
	Exercise (5 minutes)
	• Ask students to design a 4-bit adder using combinational logic.
	• Review and discuss designs.

Closure	Summarize key points: • Summarize key points: differences between combinational and sequential logic. • **Suggested Readings:** • *Digital Design and Computer Architecture* by David Harris & Sarah Harris, Chapter 3.
Evaluation	Reflective question: Why is sequential logic critical in digital systems?

Lesson Plan No. 2.1	Course Name: Modern Computer Architecture Topic: Introduction to Multi-core Architecture	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the basics of multi-core architecture. b. Learn the advantages and challenges of multi-core processors. c. Explore different multi-core design approaches.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams
Teaching Development	**Introduction (5 minutes)** • Define multi-core architecture and its significance in modern computing. • Discuss the transitionrom single-core to multi-core processors. **Development (30 minutes)** • **Multi-core Basics (15 minutes):** o Explain the concept of cores and their interconnects. o Discuss the role of multi-threading in multi-core processors. • **Challenges (15 minutes):** o Identify key challenges: synchronization, power consumption, and thermal management. • **Exercise (5 minutes)** Students diagram a simple 3-stage pipeline and identify hazards.
Closure	Summarize key points: • Summarize the benefits and challenges of multi-core architecture. • **Suggested Readings:** *Computer Architecture: A Quantitative Approach* by Hennessy & Patterson, Chapter 6.
Evaluation	Reflective question: How does pipelining improve CPU performance?

Lesson Plan No. 2.2	Course Name: Modern Computer Architecture Topic: Memory Technologies in Multi-core Systems	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Learn about different memory technologies used in multi-core systems. b. Understand the importance of memory bandwidth and latency.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Define key memory technologies: DRAM, SRAM, and emerging memories like MRAM.
	
	Development (30 minutes)
	
	• **Memory Technologies (20 minutes):**
	
	  • Explain the differences between volatile and non-volatile memory.
	  • Discuss how memory technologies impact performance in multi-core systems.
	
	• **Bandwidth and Latency (10 minutes):**
	
	Calculate memory bandwidth and discuss the impact of latency.
	
	• **Exercise (5 minutes)**
	
	Students compare different memory technologies based on speed and capacity.

Closure	Summarize key points: • Summarize memory technologies and their importance in multi-core systems. Suggested Readings: Memory Systems: Cache, DRAM, Disk by Bruce Jacob, Chapter 4.
Evaluation	Assignment: Research and report on emerging memory technologies.

Lesson Plan No. 2.3	Course Name: Modern Computer Architecture Topic: Memory Technologies in Multi-core Systems	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Learn about different memory technologies used in multi-core systems. b. Understand the importance of memory bandwidth and latency.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Define key memory technologies: DRAM, SRAM, and emerging memories like MRAM.
	
	Development (30 minutes)
	
	• **Memory Technologies (20 minutes):**
	
	  • Explain the differences between volatile and non-volatile memory.
	  • Discuss how memory technologies impact performance in multi-core systems.
	
	• **Bandwidth and Latency (10 minutes):**
	
	Calculate memory bandwidth and discuss the impact of latency.
	
	• **Exercise (5 minutes)**
	
	Students compare different memory technologies based on speed and capacity.

Closure	Summarize key points: • Summarize memory technologies and their importance in multi-core systems. Suggested Readings: Memory Systems: Cache, DRAM, Disk by Bruce Jacob, Chapter 4.
Evaluation	Assignment: Research and report on emerging memory technologies.

Lesson Plan No. 2.4	Course Name: Modern Computer Architecture Topic: The Locality Principle and Caching	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the locality principle and its role in caching. b. Explore different types of locality and their impact on cache performance.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Define key memory technologies: DRAM, SRAM, and emerging memories like MRAM.
	
	Development (30 minutes)
	
	• **Types of Locality (15 minutes):**
	
	  • Explain spatial locality (accessing nearby data) and temporal locality (reaccessing the same data).
	  • Real-time example: Access patterns in a loop.
	
	• **Caching and Locality (15 minutes):**
	
	  • Discuss how caching exploits locality for performance gains.
	
	• **Exercise (5 minutes)**
	
	Students compare different memory technologies based on speed and capacity.

Closure	• Summarize the importance of a well-designed memory hierarchy. • **Suggested Readings:** • *Computer Organization and Design* by Patterson & Hennessy, Chapter 7.
Evaluation	Reflective question: How does the locality principle improve cache performance?

Lesson Plan No. 2.5	Course Name: Modern Computer Architecture Topic: Direct Mapped Caches and Block Size	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Learn the structure and function of direct-mapped caches. b. Understand the importance of block size in cache design.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams
Teaching Development	**Introduction (5 minutes)** • Define key memory technologies: DRAM, SRAM, and emerging memories like MRAM. **Development (30 minutes)** • **Direct Mapped Caches (20 minutes):** • Explain the mapping process and how addresses are divided. • Diagram: Show a direct-mapped cache structure. • **Block Size (10 minutes):** • Discuss the trade-offs in selecting block size. • **Exercise (5 minutes)** Students calculate the hit/miss rate for a given cache configuration.
Closure	• Summarize the importance of a well-designed memory hierarchy. • **Suggested Readings:** • *Computer Organization and Design* by Patterson & Hennessy, Chapter 7.
Evaluation	Reflective question: How does the locality principle improve cache performance?

Lesson Plan No. 2.6	Course Name: Modern Computer Architecture Topic: Cache Conflicts and Associative Caches	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand cache conflicts and their impact on performance. b. Learn about associative caches and how they resolve conflicts.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	**Introduction (5 minutes)**
	• Define key memory technologies: DRAM, SRAM, and emerging memories like MRAM.
	
	Development (30 minutes)
	
	• **Cache Conflicts (15 minutes):**
	
	  • Explain what causes conflicts and how they degrade performance.
	  • Real-time example: Illustrate conflict using a specific memory access pattern.
	
	• **Associative Caches (15 minutes):**
	
	  • Explain fully associative and set-associative caches.
	  • Discuss how these caches reduce conflicts.
	
	• **Exercise (5 minutes)**
	
	Students calculate the hit/miss rate for a given cache configuration.

Closure	• Summarize the importance of a well-designed memory hierarchy. • **Suggested Readings:** • *Computer Organization and Design* by Patterson & Hennessy, Chapter 7.
Evaluation	Reflective question: Why are associative caches more effective at handling conflicts?

Lesson Plan No. 2.7	Course Name: Modern Computer Architecture Topic: Write Strategies in Caches	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Learn different write strategies in cache memory. b. Understand the trade-offs between write-through and write-back policies.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams
Teaching Development	**Introduction (5 minutes)** • Define cache write strategies and their importance in data consistency. **Development (30 minutes)** • **Write-Through vs. Write-Back (20 minutes):** • Explain the operation of write-through and write-back caches. • Discuss trade-offs: speed, complexity, and data integrity. • **Application (10 minutes):** • Compare the performance of both strategies in a multi-core environment. • **Exercise (5 minutes)** Students calculate the hit/miss rate for a given cache configuration.
Closure	• Summarize the importance of a well-designed memory hierarchy. • **Suggested Readings:** • *Computer Organization and Design* by Patterson & Hennessy, Chapter 7.
Evaluation	Reflective question: Analyze a system and recommend a write strategy.

Lesson Plan No. 2.8	Course Name: Modern Computer Architecture Topic: Advanced Cache Optimizations	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Explore advanced techniques for optimizing cache performance. b. Understand how prefetching and replacement policies enhance cache efficiency.
Teaching Aids (if any)	a. Diagrams of prefetching and replacement policies b. Simulation of cache optimization techniques

Teaching Development	**Introduction (5 minutes)**
	• Define cache write strategies and their importance in data consistency.
	
	Development (30 minutes)
	
	• **Pipelining (10 minutes):**
	
	  • Explain the stages of a pipeline.
	  • Real-time example: Simulate a pipelined instruction execution.
	
	• **Control Hazards (10 minutes):**
	
	  • Identify control hazards and discuss mitigation techniques (e.g., branch prediction).
	
	• **Cache Basics (10 minutes):**
	
	• Explain cache memory, levels, and characteristics.
	
	• **Exercise (5 minutes)**
	
	Students diagram a simple 3-stage pipeline and identify hazards.

Closure	• Summarize pipelining, control hazards, and cache basics. • **Suggested Readings:** *Computer Architecture: A Quantitative Approach* by John L. Hennessy & David A. Patterson, Chapter 3.
Evaluation	Reflective question: How does pipelining improve CPU performance?

Lesson Plan No. 3.1	Course Name: Modern Computer Architecture Topic: Introduction to Distributed Computing Systems	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the basics of distributed computing systems. b. Learn how distributed systems differ from centralized systems. c. Explore the significance of distributed computing in modern applications.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define distributed computing and its key characteristics.
	  • Discuss examples of distributed systems (e.g., cloud computing, peer-to-peer networks).
	• **Development (30 minutes):**
	  • **Distributed System Models (15 minutes):**
	  ○ Explain different models: client-server, peer-to-peer, and hybrid.
	  ○ Discuss the importance of scalability and fault tolerance.
	  • **Advantages and Challenges (15 minutes):**
	  ○ Explore the benefits of distributed computing: resource sharing, parallelism.
	  ○ Discuss challenges: synchronization, security, and data consistency.
	• **Exercise (5 minutes):**
	• Ask students to identify real-world examples of distributed systems.

Closure	Summarize key points: • Summarize the key points about distributed computing systems. • **Suggested Readings:** • *Distributed Systems: Principles and Paradigms* by Andrew S. Tanenbaum, Chapter 1.
Evaluation	Reflective question: Why are distributed systems critical for modern computing?

Lesson Plan No. 3.2	Course Name: Modern Computer Architecture Topic: Concurrency in Distributed Systems	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of concurrency in distributed systems. b. Explore how concurrency is managed in distributed environments.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define concurrency and its importance in distributed systems.
	• **Development (30 minutes):**
	  • **Concurrency Models (15 minutes):**
	  ○ Explain thread-based and process-based concurrency.
	  ○ Discuss how concurrency improves performance in distributed systems.
	  • **Challenges in Concurrency (15 minutes):**
	  ○ Identify key challenges: deadlock, race conditions, and resource sharing.
	  ○ Real-time example: Discuss a deadlock scenario in a distributed system.
	• **Exercise (5 minutes):**
	  • Students analyze a code snippet for potential concurrency issues.

Closure	Summarize key points:
	• **Closure (5 minutes):**
	  • Summarize the importance of proper concurrency management.
	  • **Suggested Readings:**
	  ○ *Operating Systems: Design and Implementation* by Andrew S. Tanenbaum, Chapter 3.

Evaluation	Reflective question: Why are distributed systems critical for modern computing?

Lesson Plan No. 3.2	Course Name: Modern Computer Architecture Topic: Concurrency in Distributed Systems	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: c. Understand the concept of concurrency in distributed systems. d. Explore how concurrency is managed in distributed environments.
Teaching Aids (if any)	c. Chalkboard/Whiteboard d. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define concurrency and its importance in distributed systems.
	• **Development (30 minutes):**
	  • **Concurrency Models (15 minutes):**
	  ○ Explain thread-based and process-based concurrency.
	  ○ Discuss how concurrency improves performance in distributed systems.
	  • **Challenges in Concurrency (15 minutes):**
	  ○ Identify key challenges: deadlock, race conditions, and resource sharing.
	  ○ Real-time example: Discuss a deadlock scenario in a distributed system.
	• **Exercise (5 minutes):**
	  • Students analyze a code snippet for potential concurrency issues.

Closure	Summarize key points:
	• **Closure (5 minutes):**
	  • Summarize the importance of proper concurrency management.
	  • **Suggested Readings:**
	  ○ *Operating Systems: Design and Implementation* by Andrew S. Tanenbaum, Chapter 3.

Evaluation	Reflective question: Why are distributed systems critical for modern computing?

Lesson Plan No. 3.4	Course Name: Modern Computer Architecture Topic: Distributed and Concurrent Programs	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Learn the principles of writing distributed and concurrent programs. b. Explore tools and techniques used in developing distributed software.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define distributed and concurrent programming and its challenges.
	• **Development (30 minutes):**
	  • **Principles of Distributed Programming (15 minutes):**
	  ○ Discuss the importance of synchronization, communication, and fault tolerance.
	  ○ Real-time example: Explain a simple distributed algorithm like leader election.
	  • **Concurrent Programming (15 minutes):**
	  ○ Explore techniques for managing concurrency: locks, semaphores, and message passing.
	  ○ Code walkthrough: Example of a concurrent program.
	• **Exercise (5 minutes):**
	  • Students modify a distributed program to handle concurrency.

Closure	Summarize key points: • Summarize the key challenges in distributed and concurrent programming. • **Suggested Readings:** • *Distributed Algorithms* by Nancy Lynch, Chapter 4.
Evaluation	Reflective question: Develop a simple distributed application with concurrency features.

Lesson Plan No. 3.4	Course Name: Modern Computer Architecture Topic: Distributed and Concurrent Programs	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: c. Learn the principles of writing distributed and concurrent programs. d. Explore tools and techniques used in developing distributed software.
Teaching Aids (if any)	c. Chalkboard/Whiteboard d. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define distributed and concurrent programming and its challenges.
	• **Development (30 minutes):**
	  • **Principles of Distributed Programming (15 minutes):**
	  ○ Discuss the importance of synchronization, communication, and fault tolerance.
	  ○ Real-time example: Explain a simple distributed algorithm like leader election.
	  • **Concurrent Programming (15 minutes):**
	  ○ Explore techniques for managing concurrency: locks, semaphores, and message passing.
	  ○ Code walkthrough: Example of a concurrent program.
	• **Exercise (5 minutes):**
	  • Students modify a distributed program to handle concurrency.

Closure	Summarize key points: • Summarize the key challenges in distributed and concurrent programming. • **Suggested Readings:** • *Distributed Algorithms* by Nancy Lynch, Chapter 4.
Evaluation	Reflective question: Develop a simple distributed application with concurrency features.

Lesson Plan No. 3.6	Course Name: Modern Computer Architecture Topic: Message Passing vs. Shared Memory Systems	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the difference between message-passing and shared-memory systems. b. Learn the advantages and disadvantages of each approach.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Introduce the two main communication models in distributed systems: message passing and shared memory.
	• **Development (30 minutes):**
	  • **Message Passing (15 minutes):**
	  ○ Explain the mechanism of message passing, its scalability, and fault tolerance.
	  ○ Real-time example: Discuss how message passing is used in microservices.
	  • **Shared Memory (15 minutes):**
	  ○ Explain shared memory systems, focusing on speed and simplicity.
	  ○ Discuss challenges like synchronization and consistency.
	• **Exercise (5 minutes):**
	  • Students compare scenarios where each communication model would be more effective.

Closure	Summarize key points: • Summarize the key points on message passing and shared memory systems. • **Suggested Readings:** • *Distributed Systems: Concepts and Design* by George Coulouris, Chapter 5..
Evaluation	Reflective question: Which communication model is better suited for a large-scale distributed system?

Lesson Plan No. 3.7	Course Name: Modern Computer Architecture Topic: Design Issues in Distributed Systems	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Explore the key design issues in distributed systems. b. Learn about the trade-offs and challenges in designing distributed systems.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Introduce the main design challenges in distributed systems.
	• **Development (30 minutes):**
	  • **Scalability and Performance (10 minutes):**
	  ○ Discuss how to design systems that scale effectively.
	  ○ Real-time example: Scalability in cloud services.
	  • **Fault Tolerance and Security (10 minutes):**
	  ○ Explore methods to ensure fault tolerance and data security in distributed systems.
	  • **Consistency and Synchronization (10 minutes):**
	  ○ Discuss the challenges of maintaining consistency across distributed nodes.
	• **Exercise (5 minutes):**
	  • Students analyze a case study of a distributed system, identifying design trade-offs.

Closure	Summarize key points: • Summarize the key design considerations in distributed systems. **Suggested Readings:** • *Designing Data-Intensive Applications* by Martin Kleppmann, Chapter 6.
Evaluation	Reflective question: How does asynchronous execution improve system responsiveness?

Lesson Plan No. 3.8	Course Name: Modern Computer Architecture Topic: Challenges in Distributed System Design	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the common challenges encountered in designing distributed systems. b. Explore strategies and solutions to mitigate these challenges.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams
Teaching Development	• **Introduction (5 minutes):** o Introduce the concept of design challenges specific to distributed systems. o Briefly list some typical challenges: latency, fault tolerance, consistency, and scalability. • **Development (30 minutes):** • **Challenge 1: Latency and Network Partitioning (10 minutes):** o Discuss the impact of latency on system performance and user experience. o Explain network partitioning and its effects on system availability. o **Solution:** Use of replication and data partitioning to mitigate latency issues. • **Challenge 2: Fault Tolerance and Reliability (10 minutes):** o Discuss the need for fault tolerance in systems where components can fail unpredictably. o Real-time example: Explain how cloud providers ensure high availability through redundancy. o **Solution:** Implement fault-tolerant protocols like Paxos or Raft. • **Challenge 3: Consistency vs. Availability (10 minutes):** o Introduce the CAP theorem and its implications on distributed systems. o Discuss the trade-offs between consistency, availability, and partition tolerance. o **Solution:** Apply eventual consistency models in scenarios where availability is prioritized. • **Exercise (5 minutes):** Present a scenario where students must identify potential challenges and suggest solutions for a hypothetical distributed system (e.g., an e-commerce platform)
Closure	**Summarize key points:** • Recap the challenges discussed and the strategies to address them. Encourage students to think about how these challenges manifest in real-world applications. **Suggested Readings:** • *Designing Data-Intensive Applications* by Martin Kleppmann, Chapter 9.
Evaluation	Reflective question: What are the key considerations when designing a distributed system that must handle large-scale failures?

Lesson Plan No. 4.1	Course Name: Modern Computer Architecture Topic: Introduction to Distributed Computing Technologies	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the fundamentals of distributed computing technologies. b. Explore various architectures and communication models used in distributed computing.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• Introduction (5 minutes):
	  • Define distributed computing technologies and their significance in modern computing.
	  • Briefly introduce different architectures: client-server, peer-to-peer, and hybrid.
	• **Development (30 minutes):**
	  • **Distributed Architectures (15 minutes):**
	  o Explain different architectures with examples (e.g., cloud computing, blockchain).
	  o Discuss the advantages and challenges of each architecture.
	  • **Communication Models (15 minutes):**
	  o Introduce communication models such as Remote Procedure Call (RPC) and Message Passing.
	  o Real-time example: Discuss how these models are used in microservices and cloud environments.
	• **Exercise (5 minutes):**
	Students identify the architecture and communication model used in a given distributed system.

Closure	Summarize key points: • Recap the challenges discussed and the strategies to address them. Encourage students to think about how these challenges manifest in real-world applications. **Suggested Readings:** • *Designing Data-Intensive Applications* by Martin Kleppmann, Chapter 9.
Evaluation	Reflective question: What are the key considerations when designing a distributed system that must handle large-scale failures?

Lesson Plan No. 4.2	Course Name: Modern Computer Architecture Topic: Clocks in Distributed Systems	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Learn the role of clocks in distributed systems. b. Understand logical and physical clocks and their importance in synchronization.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• Introduction (5 minutes):
	  • Discuss the need for clocks in distributed systems.
	  • Introduce the concept of time synchronization and its challenges.
	• **Development (30 minutes):**
	  • **Logical Clocks (15 minutes):**
	  ○ Explain Lamport□s logical clock and vector clocks.
	  ○ Real-time example: Illustrate how logical clocks are used to order events in a distributed system.
	  • **Physical Clocks (15 minutes):**
	  ○ Discuss the use of Network Time Protocol (NTP) for synchronizing physical clocks.
	  ○ Explain how physical clocks differ from logical clocks in accuracy and use cases.
	• **Exercise (5 minutes):**
	  • Students calculate the order of events using logical clocks in a given scenario.

Closure	Summarize key points: • Summarize the importance of clocks in maintaining consistency and synchronization. • **Suggested Readings:** *Distributed Systems: Principles and Paradigms* by Andrew S. Tanenbaum, Chapter 5.
Evaluation	Reflective question: Why are logical clocks preferred in certain distributed systems over physical clocks?

Lesson Plan No. 4.3	Course Name: Modern Computer Architecture Topic: Synchronization in Distributed Systems	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Explore synchronization techniques in distributed systems. b. Understand the challenges of achieving synchronization across distributed nodes.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define synchronization in the context of distributed systems.
	  • Discuss why achieving synchronization is difficult in a distributed environment.
	• **Development (30 minutes):**
	  • **Mutual Exclusion (15 minutes):**
	  ○ Explain algorithms like Lamport□s Mutex and Ricart-Agrawala for ensuring mutual exclusion.
	  ○ Real-time example: Discuss a scenario where multiple processes need to access a shared resource.
	  • **Leader Election (15 minutes):**
	  ○ Introduce the concept of leader election and discuss algorithms like Bully and Ring.
	  ○ Explain the importance of leader election in distributed systems.
	• **Exercise (5 minutes):**
	  • Students simulate a leader election process using the Ring algorithm.

Closure	Summarize key points: • Summarize the significance of synchronization in distributed systems. • **Suggested Readings:** *Distributed Algorithms* by Nancy Lynch, Chapter 3.
Evaluation	Reflective question: How does synchronization ensure consistency in a distributed system?

Lesson Plan No. 4.4	Course Name: Modern Computer Architecture Topic: Coordination and Agreement Algorithms	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the need for coordination and agreement in distributed systems. b. Learn about key algorithms used to achieve consensus and agreement.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Discuss the importance of coordination and agreement in distributed systems.
	  • Introduce common challenges like Byzantine faults and partitioning.
	• **Development (30 minutes):**
	  • **Consensus Algorithms (15 minutes):**
	  ○ Explain Paxos and Raft algorithms for achieving consensus.
	  ○ Real-time example: Discuss how consensus is achieved in blockchain networks.
	  • **Coordination Algorithms (15 minutes):**
	  ○ Discuss coordination algorithms such as Two-Phase Commit and Three-Phase Commit.
	  ○ Explain the trade-offs between reliability and performance in these algorithms.
	• **Exercise (5 minutes):**
	  • Students analyze a scenario where a distributed system must achieve consensus using Paxos.

Closure	Summarize key points: • Summarize the key concepts of coordination and agreement in distributed systems. Suggested Readings: Distributed Systems: Concepts and Design by George Coulouris, Chapter 8.
Evaluation	Reflective question: What are the key challenges in achieving consensus in a distributed environment?

Lesson Plan No. 4.5	Course Name: Modern Computer Architecture Topic: Global State in Distributed Systems	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Learn the concept of global state in distributed systems. b. Explore techniques to determine and utilize the global state.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define global state and its relevance in distributed systems.
	  • Discuss why determining global state is challenging in a distributed environment.
	• **Development (30 minutes):**
	  • **Chandy-Lamport Algorithm (15 minutes):**
	  ○ Explain the Chandy-Lamport algorithm for capturing global state.
	  ○ Real-time example: Demonstrate the use of the algorithm in a snapshot of a banking system.
	  • **Applications of Global State (15 minutes):**
	  ○ Discuss how global state is used in detecting deadlocks, monitoring, and checkpointing.
	  ○ Explain the role of global state in distributed debugging and recovery.
	• **Exercise (5 minutes):**
	  • Students simulate a global snapshot using the Chandy-Lamport algorithm.

Closure	Summarize key points: • Summarize the significance of global state in distributed systems. Suggested Readings: Distributed Systems: An Algorithmic Approach by Sukumar Ghosh, Chapter 4.
Evaluation	Reflective question: How does the concept of global state assist in managing distributed systems?

Lesson Plan No. 4.5	Course Name: Modern Computer Architecture Topic: Global State in Distributed Systems	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Learn the concept of global state in distributed systems. b. Explore techniques to determine and utilize the global state.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define global state and its relevance in distributed systems.
	  • Discuss why determining global state is challenging in a distributed environment.
	• **Development (30 minutes):**
	  • **Chandy-Lamport Algorithm (15 minutes):**
	  ○ Explain the Chandy-Lamport algorithm for capturing global state.
	  ○ Real-time example: Demonstrate the use of the algorithm in a snapshot of a banking system.
	  • **Applications of Global State (15 minutes):**
	  ○ Discuss how global state is used in detecting deadlocks, monitoring, and checkpointing.
	  ○ Explain the role of global state in distributed debugging and recovery.
	• **Exercise (5 minutes):**
	  • Students simulate a global snapshot using the Chandy-Lamport algorithm.

Closure	Summarize key points: • Summarize the significance of global state in distributed systems. Suggested Readings: Distributed Systems: An Algorithmic Approach by Sukumar Ghosh, Chapter 4.
Evaluation	Reflective question: How does the concept of global state assist in managing distributed systems?

Lesson Plan No. 5.1	Course Name: Modern Computer Architecture Topic: Introduction to High-Performance Computing (HPC) Architecture	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the fundamentals of HPC and its architecture. b. Explore the different components and configurations of HPC systems.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Brief overview of HPC, its significance, and common applications.
	  • Introduce the basic components of HPC architecture.
	• **Development (30 minutes):**
	  • **HPC Components (15 minutes):**
	  ○ Discuss the key components: processors, memory, interconnects, and storage.
	  ○ Explain the role of each component in the overall system performance.
	  • **HPC Configurations (15 minutes):**
	  ○ Overview of various configurations such as clusters, supercomputers, and grid computing.
	  ○ Real-time example: Discuss a modern HPC system used in scientific research.
	• **Exercise (5 minutes):**
	  • Identify and label the components of an HPC architecture from a provided diagram.

Closure	Summarize key points: • Summarize the importance of understanding HPC architecture. **Suggested Readings:** • *Introduction to High-Performance Computing for Scientists and Engineers* by Georg Hager and Gerhard Wellein, Chapter 1.
Evaluation	Reflective question: How do the components of HPC architecture contribute to its overall performance?

Lesson Plan No. 5.2	Course Name: Modern Computer Architecture Topic: Parallel Processing in HPC	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Comprehend the concept of parallel processing and its role in HPC. b. Learn about different types of parallelism in computing..
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define parallel processing and explain its significance in HPC.
	• **Development (30 minutes):**
	  • **Types of Parallelism (15 minutes):**
	  ○ Discuss data parallelism, task parallelism, and instruction-level parallelism.
	  ○ Real-time example: Applications of parallel processing in weather forecasting and genome sequencing.
	  • **Parallel Architectures (15 minutes):**
	  ○ Introduction to SIMD (Single Instruction, Multiple Data) and MIMD (Multiple Instruction, Multiple Data) architectures.
	  ○ Compare and contrast different parallel architectures.
	• **Exercise (5 minutes):**
	  • Identify the type of parallelism in given scenarios.

Closure	Summarize key points: • Recap the importance of parallel processing in improving computation speed. • **Suggested Readings:** • *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers* by Barry Wilkinson and Michael Allen, Chapter 2.
Evaluation	Reflective question: What are the key differences between data parallelism and task parallelism?

Lesson Plan No. 5.4	Course Name: Modern Computer Architecture Topic: Data vs. Task Parallelism	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Differentiate between data parallelism and task parallelism. b. Understand the appropriate use cases for each type of parallelism..
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Recap parallel processing and introduce the concepts of data and task parallelism.
	• **Development (30 minutes):**
	  • **Data Parallelism (15 minutes):**
	  ○ Explain data parallelism with examples such as matrix multiplication.
	  ○ Discuss when data parallelism is most effective.
	  • **Task Parallelism (15 minutes):**
	  ○ Explain task parallelism using examples like different tasks in a pipeline.
	  ○ Discuss scenarios where task parallelism is preferred.
	• **Exercise (5 minutes):**
	  • Given a problem, decide whether data or task parallelism would be more efficient and justify the choice.

Closure	Summarize key points: • Summarize the impact of memory models on parallel computing performance. • **Suggested Readings:** ○ *Parallel Computer Organization and Design* by Michel Dubois, Murali Annavaram, and Per Stenstr m, Chapter 3.
Evaluation	Reflective question: How do shared and distributed memory models affect communication and performance in parallel systems?

Lesson Plan No. 5.5	Course Name: Modern Computer Architecture Topic: High Throughput Computing	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Introduce high throughput computing (HTC) and its applications. b. Understand how HTC differs from traditional HPC...
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define high throughput computing and its importance in fields like data analysis and scientific research.
	• **Development (30 minutes):**
	  • **HTC vs. HPC (15 minutes):**
	  ○ Compare HTC and HPC in terms of tasks, execution, and performance metrics.
	  ○ Real-time example: HTC in genomics and drug discovery.
	  • **HTC Architectures (15 minutes):**
	  ○ Discuss the architecture of HTC systems including grid and cloud computing models.
	• **Exercise (5 minutes):**
	  • Identify real-world applications that benefit from HTC rather than traditional HPC.

Closure	Summarize key points: • Summarize the role of HTC in enabling large-scale scientific research. **Suggested Readings:** • *High-Throughput Next Generation Sequencing: Methods and Applications* by Robin Hesketh, Chapter 2. ○
Evaluation	Reflective question: How does HTC address different computational challenges compared to HPC?

Lesson Plan No. 5.6	Course Name: Modern Computer Architecture Topic: Vectorization in HPC	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Introduce high throughput computing (HTC) and its applications. b. Understand how HTC differs from traditional HPC...
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define high throughput computing and its importance in fields like data analysis and scientific research.
	• **Development (30 minutes):**
	  • **HTC vs. HPC (15 minutes):**
	  ○ Compare HTC and HPC in terms of tasks, execution, and performance metrics.
	  ○ Real-time example: HTC in genomics and drug discovery.
	  • **HTC Architectures (15 minutes):**
	  ○ Discuss the architecture of HTC systems including grid and cloud computing models.
	• **Exercise (5 minutes):**
	  • Identify real-world applications that benefit from HTC rather than traditional HPC.

Closure	Summarize key points: • Summarize the role of HTC in enabling large-scale scientific research. **Suggested Readings:** • *High-Throughput Next Generation Sequencing: Methods and Applications* by Robin Hesketh, Chapter 2. ○
Evaluation	Reflective question: How does HTC address different computational challenges compared to HPC?

Lesson Plan No. 5.6	Course Name: Modern Computer Architecture Topic: Vectorization in HPC	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand vectorization and its role in accelerating computations. b. Explore how vector instructions are used in modern processors.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Define vectorization and its importance in high-performance computing.
	• **Development (30 minutes):**
	  • **Vector Instructions (15 minutes):**
	  o Explain how vector instructions process multiple data points simultaneously.
	  o Real-time example: Use of vectorization in scientific computing and graphics processing.
	  • **Challenges and Optimizations (15 minutes):**
	  o Discuss common challenges like data alignment and strategies to optimize vectorized code.
	• **Exercise (5 minutes):**
	  • Analyze a given loop and suggest how it can be vectorized.

Closure	Summarize key points: • Highlight the benefits of vectorization in HPC workloads. **Suggested Readings:** • *Programming Massively Parallel Processors* by David B. Kirk and Wen-mei W. Hwu, Chapter 5. o
Evaluation	Reflective question: What types of tasks are best suited for vectorization in HPC?

Lesson Plan No. 5.8	Course Name: Modern Computer Architecture Topic: Multithreading in HPC	Course No.: COM- 401

Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of multithreading and its advantages in parallel processing. b. Explore the implementation of multithreading in HPC applications.
Teaching Aids (if any)	a. Chalkboard/Whiteboard b. Slides with circuit diagrams

Teaching Development	• **Introduction (5 minutes):**
	  • Brief overview of multithreading and its role in improving CPU utilization.
	• **Development (30 minutes):**
	  • **Multithreading Concepts (15 minutes):**
	  o Discuss the difference between single-threading, multithreading, and hyper-threading.
	  o Real-time example: Multithreading in modern processors.
	  • **Performance Considerations (15 minutes):**
	  o Explore factors that affect multithreading performance such as thread synchronization and context switching.
	• **Exercise (5 minutes):**
	  • Analyze the performance of a multithreaded vs. single-threaded application.

Closure	Summarize key points: • Highlight the benefits of vectorization in HPC workloads. **Suggested Readings:** • *Programming Massively Parallel Processors* by David B. Kirk and Wen-mei W. Hwu, Chapter 5. o
Evaluation	Reflective question: What types of tasks are best suited for vectorization in HPC?