### **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Computer Science Fundamentals\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

### **1. Computer Architecture & Organization**

#### **1.1 Introduction to Computer Systems**

* Types of Computers (Personal, Server, Supercomputers)
* Basic Components of a Computer (CPU, Memory, I/O Devices)
* Von Neumann vs Harvard Architectures
* Computer Performance Metrics (CPI, FLOPS)

#### **1.2 Digital Logic Design**

* Boolean Algebra and Logic Gates (AND, OR, NOT, XOR)
* Combinational Circuits (Adders, Multiplexers, Decoders)
* Sequential Circuits (Flip-Flops, Counters)
* Finite State Machines (FSM)

#### **1.3 Processor Design**

* Data Path Design (ALU, Control Unit)
* Instruction Set Architectures (ISA): RISC vs CISC
* Pipelining (Stages, Hazards, Forwarding)
* Superscalar, VLIW, Out-of-Order Execution

#### **1.4 Memory Hierarchies**

* Primary vs Secondary Storage (RAM, Cache, SSD, HDD)
* Cache Memory (Direct-mapped, Set-associative)
* Memory Access Time (Hit/Miss Rates, Latency, Bandwidth)
* Virtual Memory (Paging, Segmentation)

#### **1.5 I/O Systems**

* Interrupt-Driven I/O
* Direct Memory Access (DMA)
* Polling and Interrupt Handling Mechanisms
* Bus Architectures (PCI, USB, SATA)

#### **1.6 Parallel Computing Architectures**

* Multi-core and Many-core CPUs
* SIMD, MIMD, and GPU Architectures
* Cache Coherence Protocols (MESI, MOESI)
* Hyperthreading and Multithreading

**Resources:**

* *Computer Architecture: A Quantitative Approach* by Hennessy & Patterson
* Nand2Tetris course

### **2. Operating Systems**

#### **2.1 Introduction to OS**

* Types of Operating Systems (Batch, Time-Sharing, Real-time)
* OS Functions: Resource Management, Process Management
* Monolithic Kernel vs Microkernel Architecture
* System Calls (POSIX, Windows APIs)

#### **2.2 Process Management**

* Process Lifecycle (Create, Run, Wait, Terminate)
* Process Control Block (PCB) and Context Switching
* Inter-Process Communication (IPC) Methods (Pipes, Shared Memory, Message Passing)
* Signals and Handlers

#### **2.3 CPU Scheduling**

* Scheduling Algorithms (FCFS, SJF, Round Robin, Priority Scheduling)
* Multilevel Queue Scheduling
* Real-Time Scheduling (Rate Monotonic, Earliest Deadline First)

#### **2.4 Memory Management**

* Paging and Segmentation
* Contiguous vs Non-contiguous Memory Allocation
* Swapping, Thrashing
* Virtual Memory (Translation Lookaside Buffer, Page Tables)

#### **2.5 File Systems**

* File Allocation Methods (Contiguous, Linked, Indexed)
* Directory Structures (Single-Level, Two-Level, Acyclic)
* Journaling File Systems (EXT4, NTFS)
* Disk Scheduling Algorithms (FCFS, SSTF, SCAN, C-SCAN)

#### **2.6 Deadlocks and Synchronization**

* Deadlock Prevention, Avoidance, and Detection
* Banker’s Algorithm
* Synchronization Mechanisms (Mutex, Semaphore, Spinlock)
* Critical Section Problem and Solutions (Peterson’s, Lamport’s)

**Resources:**

* *Operating System Concepts* by Silberschatz, Galvin
* *Modern Operating Systems* by Tanenbaum

### **3. Kernel Programming**

#### **3.1 Introduction to the Kernel**

* User Space vs Kernel Space
* Role of the Kernel in Process Scheduling and Memory Management
* Monolithic vs Microkernel (Examples: Linux, Minix, Windows NT)

#### **3.2 Kernel Modules**

* Loadable Kernel Modules (Writing, Loading, and Unloading)
* Device Drivers: Block vs Character Devices
* Kernel Versioning and Compatibility

#### **3.3 Process and Memory Management in Kernel**

* Kernel’s Role in Process Scheduling
* Kernel Stack vs User Stack
* Memory Allocation in the Kernel (kmalloc, vmalloc)
* Paging and Page Fault Handling in the Kernel

#### **3.4 Synchronization in Kernel**

* Spinlocks, Semaphores, Mutexes in Kernel
* Interrupt Handling and Deferred Work
* Bottom Halves and Tasklets

**Resources:**

* *Linux Kernel Development* by Robert Love
* *The Linux Programming Interface* by Michael Kerrisk

### **4. CPU Architecture & Assembly Language**

#### **4.1 Machine-level Code**

* Machine Instructions and Opcodes
* Assembly Language (x86, ARM)
* Addressing Modes (Direct, Indirect, Indexed, Relative)

#### **4.2 Control Flow in Assembly**

* Conditional and Unconditional Branching
* Loops in Assembly
* Function Calls, Stack Frames, and Recursion

#### **4.3 Assembly-Level Data Manipulation**

* Bitwise Operations (Shift, Rotate, AND, OR)
* Arithmetic Operations (Add, Subtract, Multiply, Divide)
* Data Movement (MOV, PUSH, POP, LOAD, STORE)

#### **4.4 Stack Operations**

* Stack Pointer and Frame Pointer
* Function Call Conventions (Caller/Callee Saved Registers)
* Stack-based Parameter Passing

**Resources:**

* *Programming from the Ground Up* by Jonathan Bartlett
* *Assembly Language for x86 Processors* by Kip Irvine

### **5. Memory Management**

#### **5.1 Memory Hierarchies**

* Volatile vs Non-volatile Memory
* Primary vs Secondary Storage
* Cache Organization (Levels of Cache, Cache Coherence)

#### **5.2 Memory Allocation**

* Dynamic Memory Allocation (malloc, free)
* Fragmentation: Internal and External
* Heap vs Stack Allocation

#### **5.3 Virtual Memory**

* Paging, Page Tables, and TLB
* Segmentation
* Page Replacement Algorithms (FIFO, LRU, LFU)

#### **5.4 Memory Leaks and Garbage Collection**

* Memory Leaks: Detection and Prevention
* Garbage Collection Algorithms (Mark-Sweep, Reference Counting)
* Smart Pointers in C++

**Resources:**

* *The Memory Management Reference* (Online)
* *Advanced Programming in the UNIX Environment* by Stevens

### **6. Systems Programming (C)**

#### **6.1 Introduction to C Programming**

* Variables, Control Structures, Functions
* Pointers, Arrays, and Structures
* Preprocessing, Macros, and Conditional Compilation

#### **6.2 File I/O in C**

* Reading and Writing Files
* File Descriptors and Buffering
* Low-level I/O (open, close, read, write)

#### **6.3 Process Management in C**

* Process Creation and Management (fork, exec, wait)
* Signals and Signal Handlers
* Process Termination and Exit Status

#### **6.4 Threads and Synchronization**

* Introduction to Threads and Pthread Library
* Mutexes, Semaphores, and Condition Variables
* Thread Synchronization and Deadlock Prevention

**Resources:**

* *The C Programming Language* by Kernighan & Ritchie
* *UNIX Systems Programming* by Robbins and Robbins

### **7. Networking and Communication**

#### **7.1 Networking Fundamentals**

* OSI Model Layers
* TCP/IP Protocol Suite
* IP Addressing and Subnetting

#### **7.2 Socket Programming**

* TCP vs UDP Sockets
* Creating a Simple TCP/UDP Client-Server Program
* Multi-threaded Sockets Programming

#### **7.3 Network Protocols**

* HTTP/HTTPS, DNS, FTP
* ICMP, ARP, DHCP
* Network Address Translation (NAT)

#### **7.4 Network Security**

* Secure Sockets Layer (SSL) and TLS
* Firewalls and Intrusion Detection Systems
* VPNs and Secure Tunneling

**Resources:**

* *Computer Networking: A Top-Down Approach* by Kurose and Ross

### **8. Concurrency and Parallelism**

#### **8.1 Introduction to Concurrency**

* Concurrency vs Parallelism
* Threading Models: User-level vs Kernel-level Threads

#### **8.2 Thread Synchronization**

* Mutexes, Condition Variables, Semaphores
* Race Conditions and Critical Sections
* Lock-free Programming and Atomic Operations

#### **8.3 Parallel Computing**

* Multi-core Processors and Parallel Programming
* SIMD, MIMD Architectures
* OpenMP, CUDA, and GPGPU Programming

**Resources:**

* *The Art of Multiprocessor Programming* by Herlihy and Shavit

### **9. Security in Computing**

#### **9.1 Cybersecurity Basics**

* Types of Cyber Attacks (DoS, Phishing, Man-in-the-Middle)
* Firewalls and Anti-virus Software
* Cryptography Basics (Symmetric and Asymmetric Encryption)

#### **9.2 Security in Operating Systems**

* Authentication and Access Control (ACLs, Role-based Access Control)
* OS Hardening Techniques
* Secure Kernel Programming

#### **9.3 Common Vulnerabilities**

* Buffer Overflows, Format String Attacks
* SQL Injection, Cross-Site Scripting (XSS)
* Memory Leaks and Security Concerns

**Resources:**

* *Security Engineering* by Ross Anderson
* *The Web Application Hacker's Handbook* by Dafydd Stuttard

### **10. Debugging and Performance Optimization**

#### **10.1 Debugging Techniques**

* GDB: Setting Breakpoints, Stack Traces, and Core Dumps
* Memory Debugging (Valgrind, AddressSanitizer)
* Dynamic and Static Code Analysis

#### **10.2 Profiling and Optimization**

* Performance Profiling (gprof, perf)
* Cache Optimization (Data Locality, Prefetching)
* Optimizing Compilation Flags and Compiler Optimizations

**Resources:**

* *The Pragmatic Programmer* by Hunt and Thomas
* *Debugging* by David J. Agans