# IDX G10 CS H

# Study Guide Issue 2

By Gavin Yu, Edited by Ethan Qin

## Lesson 10 — Microprocessors & Memory

### RAM (Random Access Memory)

- Volatile memory: data is lost when power is off
- Stores **program instructions and data currently in use**
- Located on the **system board (motherboard)**
- CPU workflow:
  - Control Unit (CU) sends data/instructions to RAM
  - ALU processes data
  - Results are stored back in RAM before output or storage
- Uses **capacitors** to store bits:
  - Charged capacitor → 1
  - Discharged capacitor → 0

### Importance of RAM

- Acts as a **waiting room** between CPU and hard disk
- Faster access than hard disk → improves system performance
- Insufficient RAM slows down programs

### RAM Speed & Capacity

- Speed measured in **MHz** (e.g., 1066 MHz > 800 MHz)
- Capacity measured in **GB**
- Typical PCs: **2–8 GB**
- Can install more RAM up to motherboard limit

### Types of RAM

- **SRAM (Static RAM)**
  - Faster access
  - Smaller capacity
  - More expensive
- **DRAM (Dynamic RAM)**
  - Slower than SRAM
  - Larger capacity
  - Cheaper
- **SDRAM**: Synchronous DRAM, commonly used
  - Synchronize itself with system clock
  - Efficient and faster than DRAMs
- **DDR generations**: DDR1 / DDR2 / DDR3 / DDR4 / DDR5
  - Successor of SDRAM, architecture of all modern RAMs
  - Double Data Rate
    - Transfer data both on falling and rising edge of clock cycle
  - Faster with each generation
  - **Not compatible** with each other

| **Feature\Type** | **SRAM** | **DRAM** | **SDRAM** | **DDR SDRAM** |
|---|---|---|---|---|
| **Full Name** | Static RAM | Dynamic RAM | Synchronous DRAM | Double Data Rate SDRAM |
| **Storage** | Flip-flop (6 transistors | Transistor + Capacitor | Transistor + Capacitor | Transistor + Capacitor |
| **Refresh?** | No | Yes | Yes | Yes |
| **Speed** | **Fastest** | Slow | Moderate | **Fast** (2x SDRAM at same clock) |
| **Cost** | Very High | Low | Low | Low |
| **Density** | Low | High | High | High |
| **Power Use** | Higher (when active) | Lower | Lower | Lower (per gen) |
| **Primary Use** | CPU Caches | Legacy Main Memory | Old Main Memory (~1997-2000) | All Modern Main Memory |
| **Synchronicity** | Async or Sync | Async | Synchronous to Clock | Synchronous, 2x Transfer |

**Virtual Memory**

- Uses **hard disk space** to supplement RAM
- Stores inactive parts of programs/data
- Slower than RAM
- Best practice: **minimize virtual memory usage**

**ROM (Read Only Memory)**

- Non-volatile memory
- Stores **startup routines**
- Contains **BIOS (Basic Input/Output System)**
    - Performs hardware checks
    - Loads kernel into RAM from hard disc
    - **Kernel**: Core of Operating System
- **Bootstrapping (booting)**: loading OS into memory
- ROM types:
    - (Mask) MROM: factory programmed

- (Programmable) PROM: programmable once by user
- (Erasable) EPROM: erasable with UV light
- (Electrically Erasable) EEPROM: electrically erasable
- Flash memory: fast, rewritable, non-volatile

| **Feature** | **MROM** | **PROM** | **EPROM** | **EEPROM** | **Flash (NAND)** |
|---|---|---|---|---|---|
| **Programmable** | Factory only | User once | User multiple | User multiple | User multiple |
| **Erasable** | No | No | UV light | Electrically | Electrically |
| **Erase Degree** | N/A | N/A | Entire chip | Byte | Block (64KB-256KB) |
| **Write Speed** | N/A | Slow | Slow | Very slow | Fast |
| **Read Speed** | Fast | Fast | Fast | Fast | Fast (sequential) |
| **Endurance** | Unlimited | 1 write | ~100-1000 cycles | ~$10^5$ cycles | ~$10^3$-$10^5$ cycles |
| **Cost per bit** | Low (mass prod) | Medium | High | High | Very low |
| **Modern Use** | Rare | Rare | Rare | BIOS, config | Everything (SSDs, etc.) |

**Microprocessor (CPU)**

- Key features:
  - Clock speed
  - Number of cores
  - Instruction set
  - Cache size

**Factors Affecting Computer Performance**

- Clock speed (GHz)
- Front Side Bus (FSB)
- Cache
- Word size
- Instruction set
- Multi-core design
- Pipelining & parallel processing

<u>**Clock Speed**</u>

- Measured in **GHz**
- Determines number of clock cycles per second
- Performance depends on:
  - Number of instructions
  - Number of clock cycles per instruction

**FSB (Front Side Bus)**

- Transfers data between CPU and memory
- Higher FSB:
  - Transfers more data

**Cache Memory**

- Very fast, small memory inside/near CPU
- Speeds up data transfer
- Cache levels:
  - L1 Cache (core): 32-64 KB, 1-2 cycle latency
  - L2 Cache (core/shared): 256 KB - 1 MB, ~10 cycles
  - L3 Cache (shared): 8-128 MB, ~30-50 cycles
- Part of **memory hierarchy**:
  - Cache → RAM → Secondary storage

**Word Size**

- Number of bits CPU processes at once
- Larger word size = more data per cycle
- 4-bit → 8-bit → 16-bit → 32-bit → 64-bit

**Pipelining**

- Overlaps instruction stages:
  - Fetch → Decode → Execute → Store
- Improves CPU efficiency

**Parallel Processing**

- Uses multiple processors/cores simultaneously

  - Supports larger memory addressing
  - Supports more instructions
  - CPU clock = FSB speed × multiplier
- Measured in **MHz** (modern)

**Instruction Sets**

- The set of commands/operations a CPU understands.
- **RISC** (Reduced Instruction Set Computing)
  - fewer instructions, faster execution, fewer cycles
- **CISC** (Complex Instruction Set Computing)
  - complex instructions, more cycles, higher power usage
- Modern CPUs combine RISC & CISC features
- Extensions: MMX, SSE, 3DNow!, etc. (SIMD)
  - Single Instruction, Multiple Data

- Increases performance for large tasks

**Overclocking**

- Running components faster than rated speed
- Increases performance
- Risk overheating / hardware damage

**Performance = (Instructions/Program) × (Cycles/Instruction) × (Seconds/Cycle)**

↑ Code efficiency (O) ↑ IPC (architecture) ↑ Clock speed

How to optimize: *Write better codes* *Cache Hierarchy* *Pipeline, Hardware*

## Lesson 11 — Functions

### Purpose of Functions

- Complex problems → **smaller, manageable**
- Readability and reusability
- Input → process → output model

### Types of Functions

- Built-in functions
- User-defined functions
- Event procedures

### Built-in Function Examples

- int(2.6) → 2 (Number → Number)
- chr(65) → "A" (Number → String)
- ord("A") → 65 (String → Number)
- A function can have **arguments**

### Arguments

- Data passed into a function
- Can be Literals, Variables, or Expressions

### User-Defined Functions

- Have **one return value**
- Return value can be any data type
- Used like built-in functions

### Parameters

- **Formal parameters**: variables defined in function
- **Actual parameters**: values passed to function

### Parameter Passing

- **Passing by Value**
  - Used for immutable data types
  - Int, float, str, bool, tuple, range
  - Original value is not changed
- **Passing by Reference**
  - Used for mutable data types (e.g., lists)
  - List, dict, set
  - Changes affect original data

## Lesson 12 — Storage I

### Basic Components of a Data Storage System

- **Storage medium**: physical material that stores data (disk, tape, CD, DVD)
- **Drive mechanism**: reads and writes data to/from the medium

### Types of Storage Technologies

- Magnetic storage
- Optical storage
- Solid-state storage

### Magnetic Disk Storage

- Examples: **Floppy disk, Hard disk**
- Data stored as **magnetized particles**
  - One magnetic direction → 1
  - Opposite direction → 0
- Disk spins while **read/write head** accesses data
- Reading converts data into **signals**

**Hard Disk Structure**

- Made of rigid **platters** coated with magnetic oxide
- Platters grouped into a **disk pack**
- Platter has both top and bottom surfaces to record data.
- **Access arm** moves read/write heads
- Heads hover above platter surface (do not touch)
    - Each plate has its own top/bottom access arm
    - But only one in whole pack can operate
        - All heads move together
- **Head crash**:
    - Occurs if head touches platter / foreign matter
    - Causes data destruction

**Storage Capacity & Density**

- **Storage density**: amount of data stored in a given area
- Increased by:
    - Smaller magnetic particles
    - Packing particles closer
    - Layering particles
    - Vertical recording

**Physical Data Organization on Disk**

- **Track**: circular path on disk surface
    - Floppy has 80, Hard Disc > 1,000
- **Sector**:
    - Division of a track
    - Typically **512 bytes per sector**
- **Zone recording**:
    - More sectors on outer tracks
    - Improves space efficiency
- **Cluster**:
    - Group (2-8) of adjacent sectors
    - Smallest allocation unit
    - Files always occupy integer clusters
- **Cylinder**:
    - Set of tracks aligned vertically across platters
    - OS stores large files within same cylinder for efficiency
        - No need to move access arm horizontally

**Disk Access Speed**

- **Access time**: Avg time to locate and read data a piece of data
    - ~ 10 milliseconds
- Three components:
    - **Seek time**: moving access arm to

correct track
    - **Head switching**: activating correct read/write head
    - **Rotational delay**: waiting for data to rotate under head
    - Then, data is transferred to memory
- Access methods:
    - Random access (catastrophic for HDDs)
    - Sequential access (typically better)

**Data Transfer** (in Gbps)

- Transfer between disk and memory
- Performance measures:
    - Average access time (~10 ms)
    - Data transfer rate (Gbps)
- Improved using **disk caching**

**Disk Caching**

- Uses memory to store recently accessed disk data
- Reads adjacent data in advance
- Reduces disk access time
- Similar to CPU cache

**Hard Disk Controller**

- Circuit board controlling disk operations
- Interfaces disk with motherboard
- SATA, Ultra ATA, EIDE, SCSI
- Performance factors:
    - Capacity
    - Cache size
    - Rotational speed
    - Data transfer rate

## Lesson 13 — Storage II

**Optical Storage Media**

- **CD (Compact Disc)**: ~650–700 MB
- **DVD (Digital Video/Versatile Disc)**: ~2.58–4.7 GB per layer.
- **Blu-ray Disc**: ~25–27 GB per layer

**Optical Disk Storage**

- Less cost, more capacity and compactness
- Uses **laser technology** and light reflections
- Data stored as:
    - **Pits** (dark spots) → 1
    - **Lands** (light areas) → 0
- Laser reads reflected light patterns
- Single spiral track (can be ~5 km long)

**ROM, R, RW Technologies**

- **ROM**: read-only
- **R (Recordable)**:
    - Write once
    - Laser burns pits into dye
- **RW (Rewritable)**:
    - Uses phase-change crystal technology

    - Can rewrite multiple times

**Solid-State Storage (Flash Memory)**

- Stores data in erasable, electronic circuitry
- Uses transistors as gates:
    - Open → 1
    - Closed → 0
- Characteristics:
    - Non-volatile
    - Very low power usage
    - Durable
    - No moving parts
    - More expensive than HHD

**RAID (Redundant Array of Independent Disks)**

- Uses multiple disks as one unit
- Improves performance and/or reliability

**RAID 0** (Data Striping)

- Data split across multiple disks
- Faster performance (parallel read)
- Decreases reliability (more drives)

**RAID 1** (Data Mirroring)

- Data duplicated on multiple disks
- High reliability
- No performance gains

**RAID 5**

- Combines striping and parity
- Uses XOR for error recovery
- Balances performance and fault tolerance
- 4-8 drives + 1 parity drive is sweet spot
- Lets say we have 3 drives, A(10110010), B(01110110), C(11001001)
    - RAID 5 constructs a parity drive P, where P = A XOR B XOR C = 00001101
    - If drive B fails, its information can be derived from the rest of the drives
- Weakness: During rebuild time, heavy workload, if another drive fail → Disaster
    - Solution: RAID 6 (double parity)

## Lesson 14 — Input and Output Devices

**Input Devices**

- Used to enter data into computer

- Examples:
    - Keyboard
    - Pointing devices
    - Scanner
    - MICR
    - Voice input
    - Digital cameras
    - Digital video

**Keyboards**

- Types:
    - Traditional keyboard
    - Laptop keyboard
    - Wireless keyboard
    - Virtual keyboard
    - Thumb keyboard

**Pointing Devices**

- Allow interaction via physical movement
- Examples:
    - Touchpad
    - Optical mouse
    - Multitouch screen

**Touch Screen Technology**

- Touch coordinates processed like mouse clicks
- Processor compares touch position with screen image
- Types:
    - Resistive touch screen
    - Capacitive touch screen

**Output Devices**

- Provide information to user
- Output forms:
    - Graphics
    - Sound
    - Voice

## Display Technologies

- **LCD (Liquid Crystal Display)**:
  - Filters light through liquid crystals
- **LED (Light Emitting Diode)**:
  - Uses LED backlighting
- Other displays:
  - CRT
  - OLED
  - Curved monitors
  - Foldable monitors
  - E-books
  - Interactive whiteboards
  - Digital projectors

## Image Quality Factors

- Screen size (diagonal inches)
- Dot pitch (distance between dots)
- Pixel (picture element)
- Resolution (horizontal × vertical pixels)
- Color depth (bits per pixel)
- Viewing angle width
- Response rate (pixel transition speed)

## Graphics Card

- Converts CPU output into display signals
- Can be:
  - Integrated (on motherboard)
  - Dedicated (expansion card)

## GPU (Graphics Processing Unit)

- Specialized processor for graphics
- Major manufacturers:
  - Intel
  - NVIDIA
  - AMD / ATI

## Video Memory

- High-speed RAM on graphics card
- Stores image data before display
- Separate from main memory

**Ports**

- VGA: analog video
- HDMI: digital video + audio
- USB: connects peripherals
- Thunderbolt: high-speed data transfer
- Type-C: reversible connector

**Cables**

- Connect external devices to system unit
- Must match port and device connector types

## Lesson 15 – Software Basic

- **Definition of Software**
  - A step-by-step set of instructions that tells a computer how to carry out a task.
  - Referred to as a computer program.
- **Categories of Software**
  - **System Software**
    - **Operating Systems:** Manages computer hardware and software resources.
    - **Device Drivers:** Allows peripheral devices to communicate with the computer.
    - **Utilities:** Helps monitor, configure, and maintain the computer system.
    - **Programming Languages:** Used to create software.
  - **Application Software**
    - **Document Production:** e.g., word processors.
    - **Spreadsheet:** e.g., Excel.
    - **Accounting & Finance:** e.g., accounting software.
    - **Entertainment:** e.g., games.
    - **Graphics/Music/Video:** e.g., photo editors, media players.
    - **Educational Reference:** e.g., encyclopedias, tutoring software.
- **Application Software Definition**
  - Software that users apply to real-world tasks.
  - Special programming to complete special tasks.

- **Utility Software (Detailed)**
  - Designed to help monitor and configure settings for computer hardware, OS, or applications.
  - Examples:
    - Setup wizards.
    - Communications programs.
    - Security software (e.g., Antivirus, File-encryption).
    - Diagnostic tools.
    - System optimize software.
    - File Backup software.
- **Device Driver**
  - Software that helps a peripheral device establish communication with a computer.
  - Often installed via a **Device Driver Wizard**.
  - Managed through the **Device Manager** (lists hardware components like disk drives, network adapters, USB controllers).
- **Software Copyright**
  - A form of legal protection.
  - Grants the author exclusive rights to copy, distribute, sell, and modify the original work.
  - Often marked with "copyright" or "all rights reserved".
- **Software License**
  - A legal contract defining how you may use a computer program.
  - Types:
    - Single-user license.
    - Site license.
    - Multiple-user license.
    - Concurrent-use license.
- **Software Acquisition Methods**
  - **Commercial/Packaged Software:** Purchased.
  - **Demoware:** Distributed for free but limited until purchased (often pre-installed).
  - **Shareware:** Freely distributed for a trial period ("try before you buy").
  - **Free Software:** Provided free by the author; copyrighted with possible use restrictions.
  - **Open-source Software:** A variation of freeware; source code is available for viewing and modification.

Lesson 16 – Sorting

- **Sorting Problem**
  - Computers sort data via **comparison** operations (returning True or False).
  - Example Problem: Input three integers into variables a, b, c. Arrange so a is minimum, c is maximum.
    - Pseudocode solution uses **comparison** and **swap** operations.
- **Human vs. Computer Sorting**
  - **Human ("have a look"):** Flexible, creative.
  - **Computer ("compare"):** Fast, accurate, scalable.
- **Sorting Algorithm Definition**
  - An algorithm that sorts arrays of data (numbers, characters, strings, objects, etc.).
  - Defines the way to carry out **comparison** and **swap** operations.
- **Example Algorithms**
  - **Bogo Sort:**
    1. Arrange inputs randomly.
    2. Check if sorted. If yes, end. If no, repeat step 1.
  - **Insertion Sort:**
    - Builds a sorted list one item at a time by inserting each new item into its correct position.
    - **Example Process** for list [49, 38, 65, 97, 76, 13, 27, 49]:
      - Assume first element (49) is a sorted list of 1.
      - Insert 38 → [38, 49, ...]
      - Insert 65 → [38, 49, 65, ...]
      - Insert 97 → [38, 49, 65, 97, ...]
      - Insert 76 → [38, 49, 65, 76, 97, ...]
      - Insert 13 → [13, 38, 49, 65, 76, 97, ...]
      - Insert 27 → [13, 27, 38, 49, 65, 76, 97, ...]
      - Insert 49 → [13, 27, 38, 49, 49, 65, 76, 97]
    - **Code Example (Python-like pseudocode):**

```
def insertion_sort(numbers):
    for i in range(1, len(numbers)):
        for j in range(i, 0, -1):
            if numbers[j] < numbers[j-1]:
                temp = numbers[j-1]
                numbers[j-1] = numbers[j]
                numbers[j] = temp
            else:
                break
    return numbers
```

        - **Efficiency (Time Complexity):** O($n^2$) in worst/average case.
        - **Space Complexity:** O(1) (uses a constant amount of extra space).
        - **Best Case:** Input already sorted → O(n) comparisons.
        - **Worst Case:** Input reversed → ~$n^2/2$ comparisons and swaps.

- **Measuring Sorting Algorithms**
    - **Crucial Factors:**
        - **Time:** Number of comparisons and swaps.
        - **Space:** Memory required for temporary storage.
        - **Input Features:** Random, already sorted, reversed, few unique values.
    - **Complexity Analysis:** How execution time/space grows as data quantity (n) increases.
        - **O(1):** Constant time (e.g., single assignment).
        - **O(n):** Linear time (e.g., simple loop through n items).
        - **O($n^2$):** Quadratic time (e.g., nested loops, as in Insertion Sort worst case).
- **Selection Sort**
    - **Process:**
        1. Find the minimum value in the unsorted part of the array.
        2. Swap it with the first element of the unsorted part.
        3. Repeat for the remaining unsorted section.
    - **Example:** Sorting [49, 38, 65, 97, 76, 13, 27, 49] → [13, 27, 38, 49, 49, 65, 76, 97].
    - **Code Structure:** Outer loop (0 to n-2), find min in unsorted part, swap.
    - **Efficiency:** Time: O($n^2$), Space: O(1).
- **Bubble Sort**
    - **Process:**
        1. Repeatedly step through the list.
        2. Compare adjacent elements.
        3. Swap them if they are in the wrong order.

4. Repeat until no swaps are needed.

- **Example Passes** on [23,5,17,41,61,11]:
    - Pass 1: [5,17,23,41,11,61]
    - Pass 2: [5,17,23,11,41,61]
    - Pass 3: [5,17,11,23,41,61]
    - Pass 4: [5,11,17,23,41,61]
    - Pass 5: [5,11,17,23,41,61] (no swap, done)
- **Code Structure:** Outer loop (0 to n-2), inner loop compares and swaps adjacent elements.
- **Efficiency:** Time: $O(n^2)$ worst/average case.
- **Optimized Bubble Sort:** Can stop early if a pass makes no swaps (best case for sorted input: $O(n)$).

- **Common Time Complexities**
    - $O(1)$, $O(n)$, $O(n^2)$, $O(n^3)$, $O(\log_2 n)$, $O(n \log_2 n)$.
- **Algorithm Comparison & Visualization**
    - Different algorithms suit different data states (random, sorted, reversed, few unique).
    - Use animations (e.g., toptal.com sorting visualizer) to compare Insertion, Selection, Bubble, and others.