Digital Computer Organization & Design – Cheat Sheet (Theory Only)

---

1. Block Diagram of a Digital Computer

A **digital computer** consists of the following main units:

(A) Central Processing Unit (CPU)

- **Control Unit (CU):** Directs system operations.
- **Arithmetic Logic Unit (ALU):** Performs arithmetic & logic operations.
- **Registers:** High-speed temporary storage.

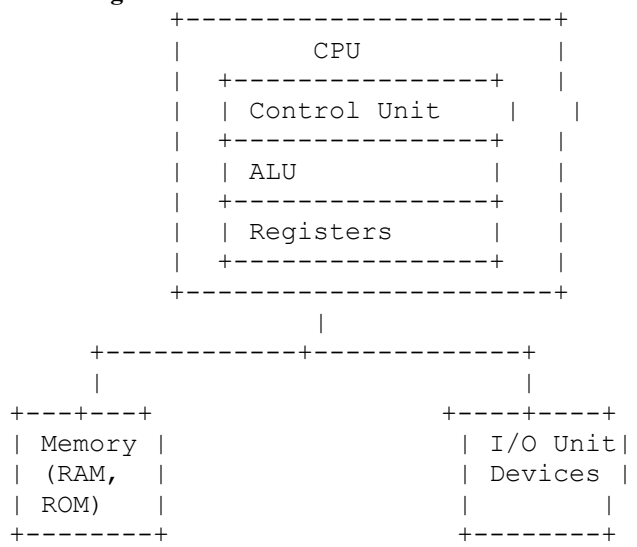
(B) Memory Unit

- **Primary Memory (RAM, ROM):** Fast access storage.
- **Cache Memory:** Stores frequently used data.
- **Secondary Storage (HDD, SSD):** Long-term storage.

(C) Input & Output (I/O) Devices

- **Input:** Keyboard, Mouse, Scanner.
- **Output:** Monitor, Printer, Speakers.

Block Diagram:



---

2. Organization & Design Concepts

Digital Computer Organization involves:

- **Instruction processing.**
- **Register operations.**
- **Memory access mechanisms.**

---

3. Instruction Codes & Registers

Instruction Code → A **binary-encoded operation** that the processor executes.
Components of an Instruction Code:

- **Opcode:** Specifies operation (e.g., ADD, LOAD, STORE).
- **Operand:** Specifies data or memory location.

Common CPU Registers:

Register	Purpose
Program Counter (PC)	Holds address of next instruction
Instruction Register (IR)	Holds current instruction
Accumulator (AC)	Stores results of operations
Memory Address Register (MAR)	Holds memory addresses
Memory Data Register (MDR)	Holds data from memory
General-Purpose Registers	Used for temporary storage

---

4. Instruction Cycle

The **sequence of steps** a CPU follows to execute an instruction.
Instruction Cycle Phases:

1st **Fetch:** Retrieve instruction from memory (PC → MAR → MDR → IR).
2nd **Decode:** Interpret instruction (CU decodes opcode).
3rd **Execute:** Perform operation (ALU processes data).
4th **Store:** Save results (back to memory or register).

Instruction Cycle Flow:

```
Fetch → Decode → Execute → Store → Next Instruction
```

---

5. Memory Reference Instructions

Instructions that access memory:

- **LOAD:** Copies data from memory to register.
- **STORE:** Copies data from register to memory.
- **ADD, SUB, MUL, DIV:** Perform arithmetic operations with memory operands.

Addressing Modes:

- **Direct Addressing:** Operand stored at given memory location.
- **Indirect Addressing:** Memory address points to another memory address.
- **Immediate Addressing:** Operand is part of instruction.

---

6. Input-Output & Interrupts

I/O Operations:

- **Programmed I/O:** CPU controls I/O transfers (slow).
- **Interrupt-Driven I/O:** Uses interrupts for efficiency.
- **DMA (Direct Memory Access):** Transfers data without CPU intervention.

Interrupts:

- **Software Interrupts:** Issued by programs (e.g., system calls).
- **Hardware Interrupts:** Triggered by external devices (e.g., keyboard, timer).

Interrupt Handling Flow:

```
Interrupt → CPU Suspends Execution → Saves State → Handles Interrupt → Resumes Execution
```

---

7. ALU Design (Arithmetic Logic Unit)

Performs:

- **Arithmetic Operations:** ADD, SUB, MUL, DIV.
- **Logical Operations:** AND, OR, XOR, NOT.
- **Shift & Rotate Operations:** Logical, Circular shifts.

ALU Components:

- **Adder/Subtractor Circuit.**
- **Comparator (for logical decisions).**
- **Control Logic.**

---

8. Execution of a Complete Instruction

Example: **ADD R1, R2** (R1 = R1 + R2)

1st **Fetch:** Instruction loaded from memory to IR.
2nd **Decode:** CU identifies opcode (ADD) & registers (R1, R2).
3rd **Execute:** ALU performs addition.
4th **Store:** Result saved back to register R1.

Multiple Bus Organization:

- **Separate buses for data, address, and control signals.**
- **Enhances parallel processing speed.**

---

9. Control Unit: Hardwired vs. Microprogrammed

Hardwired Control:

- Uses fixed logic circuits.
- **Faster but inflexible.**
- Used in **RISC (Reduced Instruction Set Computers).**

Microprogrammed Control:

- Uses **firmware-based** microinstructions.
- **More flexible but slower.**
- Used in **CISC (Complex Instruction Set Computers).**

---

10. Pipelining in CPU Execution

Basic Concept: Overlapping execution of instructions in stages.

Stages of Pipeline:

1st **Fetch → Decode → Execute → Memory Access → Write Back**

Pipeline Hazards:

- **Data Hazard:** Operand dependency between instructions.
- **Control Hazard:** Branch instruction delays.
- **Structural Hazard:** Limited hardware resources.

Techniques to Reduce Hazards:

- **Forwarding (Bypassing).**
- **Branch Prediction.**

---

11. Parallel & Vector Processing

Parallel Processing: Multiple processors work together.

- **SISD (Single Instruction, Single Data)** → Traditional CPU.
- **SIMD (Single Instruction, Multiple Data)** → Vector Processors.
- **MIMD (Multiple Instructions, Multiple Data)** → Multi-core processors.

Vector Processors:

- Process multiple data points in a single instruction.
- Used in **AI, Graphics, Scientific Computing.**

Comparison of Concepts

Feature	Hardwired Control	Microprogrammed Control
Speed	Faster	Slower
Flexibility	Less flexible	Easily modified
Complexity	More complex	Simpler design

Feature	Synchronous Counter	Asynchronous Counter
Clocking	Common Clock	Different Clocks
Speed	Faster	Slower
Delay	No ripple effect	Ripple delay occurs

Key Takeaways

Digital Computers consist of CPU, Memory, and I/O devices.
Instruction Cycle includes Fetch, Decode, Execute, Store.
ALU performs arithmetic and logic operations.
Control Unit is Hardwired (Fast) or Microprogrammed (Flexible).
Pipelining increases efficiency but faces hazards.
Parallel & Vector Processors enhance computing power.

This **Digital Computer Organization & Design Cheat Sheet** covers **CPU architecture, memory, ALU design, instruction execution, pipelining, and parallel processing**. Let me know if you need further explanations!