**Week 4-5: Home take assignment Computer Architecture and I/O Systems**

### **Part A – Theory Questions**

## **Q1. Machine Language vs. Assembly Language (10 points)**

**Machine language** is the **lowest-level programming language** consisting entirely of **binary or hexadecimal instructions** that the CPU can directly execute. It is what the hardware actually understands — for example, 10110000 01100001.  
It is hard for humans to read or write.

**Assembly language** is a **human-readable version** of machine code that uses **mnemonics (symbols)** instead of raw binary. Each assembly instruction usually corresponds to one machine instruction.  
It must be converted into machine code by an **assembler**.

**Example:**

1. Machine Language:

Example Instruction - 10110000 01100001

Meaning - Load value 97 into register AL

1. Assembly Language:

Example Instruction - MOV AL, 61h

Meaning - Move hexadecimal 61 (which is 97 in decimal) into AL

## **Q2. Instruction Format (10 points)**

| **Field** | **Meaning** |
| --- | --- |
| **OPCODE** | The **operation code** that specifies which operation to perform (e.g., ADD, SUB, LOAD). |
| **REGISTER** | The **register operand** used in the operation — a small, fast storage location inside the CPU (e.g., R1, AX). |
| **IMMEDIATE VALUE** | A **constant number** directly provided in the instruction (not stored in memory or a register). |

### **Example Instruction:**

ADD R1, #5

1. **OPCODE:** ADD → tells CPU to perform addition
2. **REGISTER:** R1 → destination register for the result
3. **IMMEDIATE VALUE:** 5 → constant value added to R1

This instruction means: Add the number 5 to the value currently in register R1 and store the result back in R1.

## **Q3. Micro-operations and Control Signals (10 points)**

1. **Micro-operations** are the **basic, low-level steps** the CPU performs on data - such as transferring data between registers, incrementing the program counter, or performing an arithmetic operation.  
   Example: PC → MAR, MDR → IR, A + B → C
2. **Control signals** are **electrical signals** generated by the CPU’s **control unit** that **initiate and coordinate micro-operations**.  
   They tell specific components (registers, ALU, memory) when to perform an action.

### **How They Work Together (Example: Fetch-Decode-Execute Cycle)**

1. **Fetch:**

Micro-ops: PC → MAR, Memory → MDR, MDR → IR, PC + 1 → PC

Control signals coordinate data transfer between registers and memory.

1. **Decode:**

Control unit interprets the instruction in IR and generates control signals for execution.

1. **Execute:**

Micro-ops: for ADD R1, R2, control signals trigger the ALU to perform R1 + R2 → R1.

Together, **micro-operations** perform the work, and **control signals** act as the “traffic lights” directing when each operation happens.

Q4. Compare RISC and CISC architectures

1. For **RISC**, a good real-world example is the **ARM processor**, which is used in most smartphones and tablets because it’s fast and energy-efficient.
2. For **CISC**, a common example is the **Intel x86 processor**, which is used in most desktop and laptop computers.