week 6 summary:

**Topics and Key Learnings**

**Lecture 6-1: Computer Fundamentals Part I**

1. **Overview of Computer Fundamentals:**

• Goal: Understanding how computers work from low-level operations to high-level programming.

• Approach:

• Introducing a simple computer architecture (“Vic”).

• Hands-on experience with machine language programming.

• Exploring the interplay between hardware and software.

2. **Vic Computer Architecture:**

• **Core Components:**

• A simple computer with a data bus, memory (RAM), and instructions.

• Instruction set includes basic operations:

• load: Load data from memory to the data register.

• store: Store data from the data register to memory.

• add and sub: Perform arithmetic using memory data and the data register.

• **Branching Control:**

• Low-level instruction goto for transferring program control.

3. **Low-Level Programming:**

• Writing programs at the machine level:

• Task: Read two numbers and write their sum using machine-level instructions.

• Focused on explicit memory and instruction handling.

4. **Fetch-Execute Cycle:**

• Step-by-step process:

• Fetch an instruction from memory.

• Decode and execute the instruction.

• Increment the program counter (PC) and repeat.

5. **Symbolic Programming:**

• Transition from numeric (low-level) to symbolic (easier-to-read) code.

• Symbolic instructions use variable names and labels for better readability.

6. **Program Translation:**

• Converting symbolic code into numeric machine code.

• Use of symbol tables to map variable names and labels to memory addresses.

7. **Example Program:**

• Task: Sum a series of numbers ending with zero.

• Process:

• Use symbolic instructions to load numbers, check conditions, and perform addition.

• Translate symbolic instructions into machine code.

8. **From Vic to a Real Computer:**

• How a simple architecture like Vic can evolve into a modern computer:

• Adding registers, memory, storage, input/output devices.

• Supporting parallel processing and multitasking.

**Key Themes:**

• **Low-Level Programming:** Understanding machine-level operations and instructions.

• **Architecture and Logic:** Exploring the building blocks of a computer and their interactions.

• **Symbolic Abstraction:** Simplifying low-level programming through symbolic representation.

• **Evolution of Computers:** Tracing the path from simple architectures to modern systems.