Week 7

a. The most important thing I learnt are about Branching, variables, loops, pointers, and I/O devices.
   - Branching allows a program to execute different parts of code based on certain conditions. It's crucial for decision making in code.
   - Variables are used to store data that can be used and manipulated throughout your code. They are fundamental to any programming task.
   - Loops allow code to be executed repeatedly until a certain condition is met. This is vital for tasks that require repetition, reducing the need for redundant code.
   - Pointers are used to store the memory addresses of other variables of data items. This is particularly used in low-level programming languages like C++.
   - I/O devices is important as they allow a program to interact with the outside world, be it a user input from a keyboard or data from file on disk.

b. How these relate to what I already know:

The concepts of branching, variables, loops, pointers, and I/O devices are extensions of my existing knowledge of C++ programming that I did earlier. I can relate to each concept of the C++ to this assembly language which makes it easier for me to understand this module. They're the building blocks of code, allowing for complex, efficient, and interactive programs.

c. My course team wants me to learn these concepts because they are fundamental to programming. Understanding them will enable me to write efficient code, solve complex problems, and build robust software applications. It's part of equipping us with comprehensive skills for your future career in technology.

Week 8

a. some of the most important thing I learnt in this week are:

- Von Neumann Architecture: A computer architecture that consists of a CPU, a memory, and an input/output unit, connected by buses.
- Information Flow: How data, addresses, and control signals are transmitted between the key components of the Von Neumann Architecture.
- Fetch-Execute Cycle: The basic CPU loop that fetches an instruction from the instruction memory, decodes it, and executes it.

b. Most of the things are new to me especially Von Neuman Architecture. But still I can relate the Fetch-Execute cycle with the C++ coding I learnt earlier. Concepts like compilation, execution, and instruction handling can be closely related to both high-level programming and this assembly language. Understanding this can help you write more efficient code, as you'll have a better idea of what your code is doing at the hardware level.

c. This knowledge can be useful when optimizing code, debugging performance issues, or designing new hardware. Understanding that each loop iteration or function call corresponds to a series of machine instructions can make us more aware of the cost of these operations. These concepts are also the foundation for more advanced topics in computer science, such as operating systems, compilers, and parallel computing.

Week 9

a. The most important thing I learnt this week is how to implement the Hack computer, a simple 16-bit machine that can run programs written in the Hack assembly language. then I learnt how to write an assembler, a software that translates Hack assembly code into Hack binary code that can be executed by the Hack computer. Finally, learnt how to handle symbols in Hack assembly code, such as pre-defined symbols, label symbols, and variable symbols.

b. I can relate the hack assembler to the binary number conversion that I did back in mathematics. The similar operation happens right here as well. It translates the low-level language into a machine language which the computer can understand. We can think of it as a translator or interpreter that converts human instructions into computer commands. We can see that a different number of arithmetic and logical operations are implemented using binary codes and logic gates.

c.

- By learning about the Hack computer, I gain a fundamental understanding of how a computer works at the hardware level. This includes knowledge about memory, CPU, and how they interact.

- Learning Assembly Language can give me a better understanding of how high-level languages (like Python or Java) are translated into instructions that a computer can execute.

- The knowledge gained from this course can be applied in various fields, such as software development, hardware engineering, and more. It provides a strong foundation for further studies in computer science.

Week 10

a. Some of the most important thing I learnt this week are:
   - How to implement a Hack assembler using high-level programming language, following a modular design and a staged development approach.
   - How the assembler translates assembly code into machine code by parsing, translating, and handling symbols, using the Hack language specification and the symbol table.
   - How the CPU executes machine code instructions by fetching, decoding, and executing them, using the registers, ALU, buses, memory, and program counter.
   - How the CPU hardware is built from logic gates using the NAND gate as the universal gate, and how the logic gates operate on binary numbers.

b. I can relate to my previous knowledge of logic gates and hardware, where CPU hardware is built from the logic gates using the NAND gates as the universal gate can be a direct application of that knowledge. I can also relate to how high-level languages are translated into machine code that the CPU can execute which I learnt from previous weeks. At the end it is mentioned that NAND gate is a universal gate, I remember studying the same thing from the first few weeks of this course commencement. Now I can see why it was mentioned like that after coming to know its importance.

c. My course team wants me to learn this because:
   - By learning how an assembler works, I can gain a practical understanding of computer architecture. You see how high-level languages are translated into machine code that the CPU can execute.
   - The modular design and the staged development approach of the Hack Assembler provide a real-world example of good software engineering principles. It shows how complex problems can be broken down into manageable parts.
   - The topics covered in this week, such as logic gates, the stored program concept, and the fetch-execute cycle, builds upon and deepen our understanding of previous topics in the course.