Module 1 – Unit 1 Class Notes

Introduction to Computers and Programming Languages

At a Glance

- Overview
- Objectives
- Additional Resources
- Key Terms

Lecture Notes

Overview

In Module 1, Unit 1, you will learn the main components of a computer system, the history and evolution of computer languages, and some fundamental ideas about how to solve problems with computer programming.

Unit Objectives

In this unit, you will:

- Learn about different types of computers
- Explore the hardware and software components of a computer system
- Learn about the language of a computer
- Learn about the evolution of programming languages
- Examine high-level programming languages
- Discover what a compiler is and what it does
- Examine how a Java program is processed

Unit Tips

Introduction

1. Through computers, the technological revolution is drastically changing the way we live and communicate with one another.

2. This is all made possible through the use of software, which are computer programs.

3. These programs are created with the help of programming languages, and Java is an example of one such language.

Elements of a Computer System

1. A computer is an electronic device capable of performing commands.

2. The basic commands that a computer performs are input (get data), output (display result), storage, and performance of arithmetic and logical operations.

Hardware

1. The major hardware components includes the central processing unit (CPU), the main memory (MM), the input/output devices, and secondary storage.

2. The main memory is also called random access memory (RAM).

Central Processing Unit and Main Memory

1. The central processing unit (CPU) is the brain of the computer and the single most expensive piece of hardware in a computer, and the more powerful the CPU, the faster the computer.

2. The main components of the CPU are the control unit (CU), the arithmetic logic unit (ALU), and the registers.

3. Main memory is directly connected to the CPU and all programs must be loaded into main memory before they can be executed. All data must be read into main memory before a program can manipulate it.

4. Main memory is also referred to as random access memory, or RAM, and when the computer is turned off, everything in main memory is lost.

5. Main memory is an ordered sequence of memory cells, and each cell has a unique location in main memory, called the address of the cell.

Secondary Storage

1. Information stored in main memory must be transferred to secondary storage for longer-term storage.

2. Some examples of secondary storage are hard disks, floppy disks, ZIP disks, CD-ROMs, and tapes.

Input/Output Devices

1. Input devices feed data and programs into computers. Examples of input devices are keyboard, mouse, and scanner.

2. The computer uses output devices to display and store results. Examples of output devices are screen, printer, and speakers.

Software

1. Software consists of programs written to perform specific tasks, and there are two types of programs: system programs and application programs.

2. System programs control the computer, and the operating system is the system program that loads first when you turn on your PC.

3. The operating system monitors the overall activity of the computer and provides services.

4. Application programs perform a specific task, and the operating system is the program that runs application programs.

Language of a Computer

1. Electrical signals move along channels inside the computer, and there are two types of electrical signals: analog and digital.

2. Digital signals are more reliable carriers of information, and therefore, the computer uses digital signals.

3. Because digital signals are processed inside a computer, the language of a computer, called machine language, is a sequence of 0s and 1s.

4. Every character on the keyboard is encoded as a sequence of bits (0 or 1). The most commonly used encoding scheme on personal computers is the seven-bit American Standard Code for Information Interchange (ASCII).

5. The ASCII data set consists of 128 characters numbered 0 through 127, wherein each character requires a byte of storage, and it is a subset of **Unicode**, which consists of 65,536 characters, wherein each character requires two bytes of storage.

Evolution of Programming Languages

1. Most basic computer language is machine language and machine language provides program instructions in bits.

2. The designers of different CPUs may choose different sets of binary codes to perform the computer's operations, and thus the machine language of one computer is not necessarily the same as the machine language of another computer.

3. Assembly languages were developed to make the programmer's job easier. In assembly language, an instruction is an easy-to-remember form called a mnemonic.

4. A program called an assembler translates the assembly language instructions into machine language.

5. Languages like Basic, FORTRAN, COBOL, Pascal, C, C++, C#, and Java are all high-level languages.

6. Java instructions first need to be translated into an intermediate language called bytecode and then interpreted into a particular machine language.

7. A compiler is a program that translates a program written in a high-level language into the equivalent machine language, in the case of Java, the machine language is bytecode.

8. Different types of CPUs use different machine languages, and to make Java programs machine independent, the designers of Java introduced the Java Virtual Machine (JVM).

Processing a Java Program

1. The source program is created in a text editor (like TextPad) and must be saved in a file with the same name as the class.

2. After saving the source program, it must be compiled. The compiler checks for the correct syntax and translates the program into bytecode.

3. To run a Java application, the `.class` file must be loaded into computer memory.

4. The programs that you write in Java language are usually developed using an Integrated Development Environment (IDE).

5. Java application consists of a collection of classes, and in order to run the bytecode, the classes must be connected by the loader, which loads the bytecode into main memory.

6. As the classes are loaded into main memory, the bytecode verifier verifies that the bytecode for the classes is valid, and then an interpreter translates each bytecode instruction into the machine language of your computer.

7. The interpreter translates and executes one bytecode instruction at a time.

Programming with the Problem Analysis-Coding-Execution Cycle

1. Programming is a process of problem solving, and an algorithm is a step-by-step problem-solving process in which a solution is arrived at in a finite amount of time.

2. Analyzing the problem is the first and most important step.

3. The next step is to design an algorithm to solve the problem. If you break the problem into subproblems, you need to design an algorithm for each subproblem. Finally, each algorithm must be checked for correctness.

4. The next step is to convert the equivalent code into a high-level language, and verify the syntax by running the code through the compiler.

5. When all the syntax errors are removed, the compiler generates the machine code (bytecode in Java), and the program can be executed.

Programming Methodologies

This section outlines two popular programming approaches.

Structured Programming

1. Dividing a problem into smaller subproblems is called structured design, and implementing a structured design is called structured programming.

2. Structured design approach is also known as top-down design, bottom-up design, stepwise refinement, and modular programming.

Object-Oriented Programming

1. In object-oriented design (OOD), the first step in the problem-solving process is to identify components called objects, which form the basis of the solution, and to determine how these objects interact with one another.

2. The next step is to specify the relevant data for each object and possible operations to be performed on that data.

3. An object consists of data and the operations on those data. In OOD, the final program is a collection of interacting objects, and a programming language that implements OOD is called an object-oriented programming (OOP) language.

4. To create operations, you write algorithms and implement them in a programming language.

5. A data element in a complex program usually has many operations, and to separate operations from each other and to use them effectively and in a convenient manner, you use methods to implement algorithms.

6. In Java, the mechanism that allows you to combine data and operations on the data into a single unit is called a class.

Additional Resources

1. Ada Augusta, Countess of Lovelace:
   www.ideafinder.com/history/inventors/lovelace.htm

2. More information on the history of computers:
   www.eingang.org/Lecture/

3. An overview of the Java programming language:
   http://java.sun.com/docs/overviews/java/java-overview-1.html

4. More information about computer hardware:
   www.infosyssec.net/infosyssec/comphard.htm

5. More information about Unicode:
   www.unicode.org/

6. An overview of a number of programming methodologies:
   http://oopweb.com/CPP/Documents/Intro2OOP/VolumeFrames.html?/CPP/Documents/Intro2OOP/Volume/node3.html

Key Terms

- **Address –** The unique location of each cell in the main memory.
- **Algorithm –** A step-by-step problem-solving process in which a solution is arrived at in a finite amount of time.
- **American Standard Code for Information Interchange (ASCII) –** The most commonly used encoding scheme on personal computers.
- **Analog signals –** Continuous waveforms used to represent things, such as sound.
- **Application programs –** Programs that perform specific tasks, such as word processors, spreadsheets, and games.
- **Arithmetic logic unit (ALU) –** The component of the CPU that carries out all arithmetic and logical operations.
- **Assembler –** A program that translates the assembly language into machine language.
- **Assembly language –** Renders instructions into mnemonics to make the programmer's job easier.
- **Base 2 –** The number system computers use in which all numbers are represented by sequences of 0s and 1s.
- **Base 10 –** The number system used in our daily lives in which all numbers are represented by sequences of 0s through 9s.
- **Binary code –** A sequence of 0s and 1s.
- **Binary number –** A sequence of 0s and 1s.
- **Binary digit –** The digit 0 or 1.
- **Bit –** An abbreviation of the term "binary digit," the binary digit 0 or 1.
- **Bottom-up design –** see **structured design**
- **Byte –** A sequence of eight bits.
- **Bytecode –** The intermediate language that Java is translated into by the Java interpreter.
- **Central processing unit (CPU) –** The brain of the computer and the single most expensive piece of hardware in your personal computer.
- **Class –** In Java, the mechanism that allows you to combine data and operations on the data into a single unit.
- **Compiler –** A program that translates instructions written in Java into bytecode.
- **Decimal system –** The number system humans use in their daily lives.
- **Digital signals –** A representation information with a sequence of 0s and 1s, in which 0 represents a low voltage and 1 represents a high voltage.
- **High-level languages –** Programming languages that more closely resemble spoken languages.

- **Input devices –** Devices that feed data and programs into computers, such as a keyboard or mouse.
- **Integrated Development Environment (IDE) –** A software package that facilitates programming and contains many programs that are useful in creating programs written in Java.
- **Interpreter –** A program that reads, translates each bytecode instruction into the machine language of your computer, and then executes it.
- **Java Virtual Machine (JVM) –** A hypothetical computer introduced by Java that uses bytecode and allows Java to be platform independent.
- **Kilobyte (KB) –** 2^{10} or 1024 bytes.
- **Loader –** A program that connects the bytecode for the classes used in a Java application and transfers executable code into main memory.
- **Machine independent –** The ability to run on many different types of computer platforms.
- **Machine language –** The most basic language of a computer.
- **Main memory –** Computer memory that is connected directly to the CPU, into which all programs must be loaded before they can be executed. Also called Random Access Memory, or RAM.
- **Memory cells –** The ordered sequence of cells that comprise main memory.
- **Methods –** A sequence of statements or instructions whose objective is to accomplish a task.
- **Mnemonics –** Easy-to-remember instructions.
- **Modular programming –** see **structured design**
- **Object-oriented design (OOD) –** A widely used programming methodology, in which program components called objects are identified and form the basis of the solution.
- **Object-oriented programming (OOP) –** A programming language that implements OOD.
- **Objects –** Components of a program that combine data and operations on data into a single unit.
- **Operating system –** The system program that loads first when the PC is turned on.
- **Output devices –** Devices that the computer uses to display and store results, such as the monitor, printer, or secondary storage.
- **Pseudocode –** An "outline" of a program that could be translated into actual code.
- **Secondary storage –** A device that stores long-term information, such as hard disks, floppy disks, flash memory, CD-ROMs, and tape drives.
- **Source program –** A program written in a high-level programming language.
- **stepwise refinement—**see **structured design**
- **Structured design –** A program methodology in which a problem is divided into smaller subproblems and then analyzed and solved.
- **Structured programming –** The process of implementing a structured design.
- **System programs –** The programs that control the computer.
- **Top-down design –** see **structured design**
- **Unicode –** A way to represent 65,536 characters as a sequence of 16 bits.