KOLEJ MATRIKULASI LABUAN
LABUAN MATRICULATION COLLEGE

TOPIC 1

INTRODUCTION TO PROGRAMMING

COMPUTER SCIENCE II | SC025

Learning Outcome

At the end of the lesson, students should be able to:

a) Define programming language, programming paradigm and language translators.

b) Differentiate paradigms of programming language: procedural, object oriented and logic.

c) Define types of language translator: compiler, interpreter and assembler.

Introduction

Computer Program - Series of instructions for a computer to execute.

How do we make a **computer program**?
We use programming languages to write computer programs.

Why do we need **programming languages**?
Programming languages are designed for computers to be able to understand and execute its instructions.

Programming Languages

Definition:

Programming language is a set of words, abbreviation, and symbols that enables a programmer to give instructions to a computer.

Examples: Java, C, C++, JavaScript, Python, HTML, etc...

Different programming languages have different rules, syntax and functions.

Types of Programming Language

Low-level languages

- A type of programming language that contains basic computer instructions that the computer can understand and execute.
- Communicates directly with hardware.
- Mostly used in operating systems or programs that require direct hardware access.

Machine Language

- Set of instructions and data in the form of binary numbers: “0” or “1”
- Hard for humans to understand.
- Easy for computers to understand.
- First-generation programming language.
- Machine dependent. Program on one computer will not work on another computer.

Assembly Language

- Contains basic English commands or abbreviations.
- Easier for humans to understand compared to machine language.
- Needs to be translated into machine language before it is run by the computer.
- Second-generation programming language.

```
                    EQU     %00010011
                EG  EQU     %00010001

                INITA   LDA A   #RESETA   RESET ACIA
                    04          STA A   ACIA
                    1           LDA A   #CTLREG   SET 8 BITS AND 2 STOP
                    80 04       STA A   ACIA

                7E C0 F1        JMP     SIGNON    GO TO START OF MONITOR

                        ****************************************
                        * FUNCTION: INCH - Input character
                        * INPUT: none
                        * OUTPUT: char in acc A
                        * DESTROYS: acc A
                        * CALLS: none
                        * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04   INCH    LDA A   ACIA      GET STATUS
C013 47                 ASR A             SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC     INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A   ACIA+1    GET CHAR
C019 84 7F              AND A   #$7F      MASK PARITY
C01B 7E C0 79           JMP     OUTCH     ECHO & RTS

                        ****************************************
                        * FUNCTION: INHEX - INPUT HEX DIGIT
                        * INPUT: none
                        * OUTPUT: Digit in acc A
                        * CALLS: INCH
                        * DESTROYS: acc A
                        * Returns to monitor if not HEX input
```

Comparison between Machine and Assembly language

	Machine Language	Assembly Language
Complexity to humans	Cannot be easily understood by humans	May be understood by humans
Complexity to computers	Computers can directly execute	Needs to be translated using an assembler
Characteristic	Made up of 0s and 1s (binary)	English-like abbreviations and commands

High-level languages

- These languages use English-like words that are closer to human language.
- Easy to read, write, and maintain. Programmer-friendly.
- Portable as these languages can run on different computers.
- Needs to be translated into machine code before it is executable by the computer.

Comparison between High and Low-level Languages

Low-level Language	High-level Language
Machine-friendly language; Harder to be understood and learnt by humans	Programmer-friendly language; Easily understood and learnt by humans
Non-portable; Machine dependent	Portable; Machine independent
Consists of binary digits or English-like abbreviations	Consists of English-like words
Memory efficient	Less memory efficient

Programming Paradigm

Definition:

Programming paradigm is an approach to solving problems using programming language.

The different styles by which a computer language looks at the problem to be solved.

There are three (3) main types of programming paradigms:

Procedural Programming

Object-oriented Programming

Logic Programming

Procedural Programming

- Procedural programming is a structured approach to software development.
- Programs are organized into procedures to perform computations.
- A procedure is a series of instructions to be carried out by the computer step-by-step.
- Top-down approach; Linear.
- Examples: C, COBOL, Python.

Object-oriented Programming

- Object-oriented programming (OOP) that revolves around the concept of objects.
- An object in an item that can contain both data and procedures that read and manipulate that data.
- Bottom-up approach;
- OOP allows programmers to reuse and modify existing objects.
- Examples: Java, C++, JavaScript, C#

Logic Programming

- Logic programming revolves around program statements that express facts and rules about a problem within a system of formal logic.
- Rules are written in logical clauses, “X is true if Y and Z are true”
- Facts are expressed without a body, “X is true”.
- Examples: Prolog, ASP, Datalog
- Also largely used is Artificial Intelligence (AI) models.

Comparison of Programming Paradigms

Procedural	Object-oriented	Logic
Top-down approach	Bottom-up approach	No explicit algorithm
Focuses on procedures and data	Focuses on objects and methods	Focuses on facts and rules
C, COBOL, Python	C++, Java, JavaScript	Prolog, ASP, Datalog

Language Translator

Definition:

A translator is a computer program that translates programming source code into machine language (machine code).

There are three (3) types of language translators: assembler, comipler, and interpreter.

Assembler

- Assembler converts programs written in assembly language into machine code.

```
0        LDR R0, 150
1        LDR R1, 151
2        ADD R2, R1, R0
3        CMP R2, #10
4        BLT end
5        MOV R2, #10
  end:
6        STR R2, 152
7        HALT
```

Assembly language

Assembler

Machine code

Assembler

- Recap: Assembly language is a low-level programming language that uses English-like abbreviations to represent instructions / code.
- The assembler converts assembly language source program into machine language to create an executable program called object code.
- However, assemblers are processor-dependent and each kind of computer may be programmed in its own assembly language.

Compiler

- Compiler converts entire high-level source code into machine language.

```
<! DOCTYPE  html>
<html>

<head>
   <title>Page Title</title>
</head>
<body>
   <h2>Heading Content</h2>
   <p>Heading Content</p>
</body>

</html>
```

High-level programming language

Compiler

Machine code

Compiler

- Recap: High-level programming languages uses English-like words to represent instructions / code.
- A compiler converts high-level source code into an executable file containing machine code (object code).
- Error reports are generated after the entire program is compiled, offering a comprehensive overview of potential issues.

Interpreter

- An interpreter translates high-level source code into machine code and executes it.
- This does not produce an executable file.
- Each line of source code is read then converted into machine language, then executed. Any error is immediately detected and shown.

```
System.out.print("hello");
```

→

```
0111000101010
```

→ Execution of machine code

Line of high-level source code

Machine code

Language Translators

Assembler	Compiler	Interpreter
Source program written in assembly language	Source program written in high-level programming language	Source program written in high-level programming language
Converts into machine language	Converts into machine language (object code)	Converts into machine language AND executes it
Creates executable file	Creates executable file	Does NOT create executable file