Research Analysis

Topic: Programming Languages

★ Low-Level Languages

1. Machine Language (1st Gen)

- Instructions are in binary (0s and 1s)
- Machine-specific
- Difficult for humans to read and write.

2. Assembly Languages (2nd Gen)

- Uses mnemonics (codes) to represent machine instruction
- Easier to read and write as compared to machine lang.
- Machine-specific.
- Requires an assembler.

Example: MOV AX, BX (moves content from BX to AX)

★ High level Languages.

1. 3rd Gen. (C, C++, Java, ~~Ph~~ Python, Javascript etc)

- Closer to human language
- Machine independent
- Requires a compiler or interpreter to translate high-level code to machine code
- Offer features like d.s., control strs & functions

Example: (C++): int sum = a + b;

2. 4th Gen

- Designed for specific tasks (e.g. Database management, AI, Scientific computing)
- Often declarative (describe desired outcome rather than steps to achieve it)

★ Middle - Level Languages.

- C, C++
  - → Combine features of both low-level and high-level la
  - → Offer some control over hardware (like memory management) while providing high level abstraction.
  - → Suitable for system programming (operating sys., driver

⇒ Comparison:

Feature	Low - Level (Machine / Assemb)	Middle (C / C++)	High Level (Java, Python etc.)
~~Abstraction~~			
Abstraction	Very low	moderate	High
Machine depend.	High	Moderate	Low
Readability	Difficult	moderate	Easy
Portability	Low	moderate	High
Exec. speed.	Very fast	fast	moderate to slow
Mem. Control	Direct	Some	Limited

* Concepts about programming languages.

Feature	Compilation	Interpretation
Translation Process	Translates entire source code into machine code before exec.	Translates and executes source code line-by-line.
Execution Speed.	Faster execution since entire code is already in machine code.	Slower execution due to overhead of translation during runtime.
Debugging	Debugging can be more difficult as entire code needs to be recompiled after changes	Easier as errors are detected line-by-line.
Portability	Compiled code is platform specific. Requires recompilation for diff. platforms	Interpreted code is more portable as the interpreter handles platform-specific details
Memory Usage	Compiled code generally requires more memory	Interpreted code can be more mem-effecient as it only needs to load and exec. 1 line at a time.
Examples.	C, C++, Java (Just-In-Time compilation is a hybrid approach)	Python, Javascript, Ruby

✱ Role of Assemblers, Compilers and Interpreters.

1. Assemblers:

- Translate assembly lang. code into machine code.
- 1 to 1 correspondence: Each assembly lang ins. typically corresponds to a single ins.

2. Compliilers:

- Translate high-level lang. code into machine code or an intermediate representation (like bytecode)
- Process of compilation:
  - a. Lexical Analysis: Converting source code into a stream of tokens.
  - b. Syntax Analysis (Parsing): Checking grammatical structure of code (indents, end operators like semicolon etc.)
  - c. Semantic Analysis: Checks the meaning and validity of code.
  - d. Intermediate Code Generation: Creating an intermediate representation of code.
  - e. Optimization: Improves efficiency of generated code.
  - f. Code Gen.: Generates final machine code or bytecode
- Output: An executable file that can be ~~re~~run directly on target machine (or vm. for byte code ex. Java)

3. Interpreters.

- Translate and execute high-level code line by line.
- No seperate compilation phase: Code is translated and exec. simultaneously.

- Process : The interpreter reads each line of code, translates it into an intermediate representation (or directly executes it) and proceeds to next line.
- Slower execution compared to compiled code: Due to overhead of translation during runtime.
- Advantages : Easier debugging, better portability. (it not handles platform - specific details)

⇒ Comparison

Feature	Assembler	Compiler	Interpreter
Input Language	Assembly Language	High-level Language	High-level Language.
Output	Machine code	Machine code / Bytecode	Direct Execution
Translation Process	1-1 instruction mapping	Translates entire program	Translates and exec. line by line.
Execution Speed	Fast	Fast	Slower
Debugging	More difficult	More difficult	Easier
Portability.	Low	Moderate	High.