COP2270 – C for Engineers

Fall 2025 – Independent Study Assignment
Instructor: Prof. Lester D. Suarez
Email: lsuarez9@mdc.edu
Due Date: 12/11/2025
Submission Format: Typed report (PDF or Word, 4–6 pages)

Assignment Title:
Understanding Computer Architecture, Software Fundamentals, and Programming Languages

---

Purpose
This independent study is designed to strengthen your understanding of how **computer systems**, **software tools**, and **programming languages** relate to C programming and the engineering problem-solving process. You'll investigate how hardware and software interact and explore programming language characteristics to make informed choices in future projects.

Instructions
You must address the **three sections** below using clear explanations, illustrations, and comparisons where relevant. Each section must be labeled. You may use diagrams, tables, or charts to support your explanations.

Section 1: Computer Hardware Fundamentals
Address the following:

1. Describe the architecture and operation of a typical computer system (e.g., CPU, memory, input/output).
2. Draw and label a block diagram showing input, output, CPU, storage, and main memory.
3. Identify key storage types:
    - Registers
    - Main memory
    - Secondary storage
      Discuss their advantages and disadvantages (e.g., speed, volatility, cost).
4. Explain the purpose and characteristics of:
    - Stack
    - Heap
    - Code Segment
    - Data Segment

Section 2: Software Fundamentals

Address the following:

1. Draw a block diagram of the C **compilation process** (include: source code → compiler → assembler → linker → executable).
2. Explain the difference between **static** and **dynamic linking** of libraries, including pros/cons.
3. Discuss why a C program compiled on one operating system (e.g., Windows) may not run on another (e.g., Linux).
4. Describe how to use a **C language build system** (such as GNU Make) to compile a multi-file project.
5. Briefly compare the **compiling and linking process** in POSIX (Linux/macOS) vs. Microsoft Windows environments.

Section 3: Programming Language Comparison

Address the following:

1. Compare:
    - High-level vs. low-level languages
    - Managed vs. unmanaged code
    - Interpreted vs. compiled languages
2. Compare **at least five** of the following languages:
    - Assembly, C, C++, MATLAB, Java, Python, C#, Perl, PHP
      Discuss differences in: execution model, syntax, memory management, and typical use cases.
3. Based on different applications (e.g., embedded systems, web development, engineering simulation), **recommend the most suitable language** and justify your selection.

Submission Requirements

- Report must be 4–6 pages (excluding diagrams/tables)
- Include citations and a reference section in APA or MLA format
- Submit via email to lsuarez9@mdc.edu or GitHub to the course repo under: https://github.com/lsuarez9/COP2270/<Student_Folder>
- File name: LastName_FirstName_COP2270.pdf

Grading Rubric

Criteria	**Points**
Section 1: Hardware Fundamentals	25
Section 2: Software Compilation Concepts	25
Section 3: Language Comparison & Evaluation	25
Structure, clarity, and depth of analysis	15
Visuals (block diagrams/tables/charts)	5
Proper citations and formatting	5
Total	**100**

Recommended Tools

- GCC Compiler (Linux or Windows)
- Visual Studio Code or Code::Blocks
- Online resources: cplusplus.com, gnu.org, Wikipedia

Reminder

This assignment supports course outcomes related to:

- Data representation, software development, and toolchains
- Program compilation and memory models
- Programming language comparisons for engineering applications