Course Title: Computer Architecture
Course Code: CSE360
Section 1

Assignment – Multiplication Game (Computer Architecture Focus)

Due Date: May 19, 2025

Project Overview:

For this semester's **Computer Architecture** course, your term project involves implementing a **Multiplication Game**in **C or C++**, focusing on how software interacts with and simulates low-level architectural principles.

The game idea is inspired by the original version here:
🔗 https://www.mathsisfun.com/games/multiplication-game.html

This project will encourage you to simulate concepts typically addressed at the hardware or instruction-level using C/C++, such as:

- Efficient use of memory and stack/heap
- Bitwise and arithmetic operations
- Logical control structures that mirror instruction cycles
- Simulation of registers and memory layout (optional for extra credit)

Key Architecture-Related Goals:

1. **Arithmetic and Logic Simulation:**
   - Use **bitwise operators**, **modular arithmetic**, and control flow to simulate operations close to machine level.
2. **Memory Organization:**
   - Use pointers, arrays, and structures to manage data efficiently.
   - Reflect on how the game state is stored in memory (e.g., board representation, player scores, etc.).
3. **Instruction-Level Logic:**
   - Implement decision-making and game logic using control flow constructs (e.g., loops, `if-else`, `switch`) that mimic basic instruction execution paths.
4. **I/O Simulation:**
   - Use standard input/output to handle interactions and mimic how a system interfaces with external input devices.

5. **Performance Awareness:**
    - Encourage optimized logic using efficient algorithms and minimal memory overhead, reflecting principles of low-level system design.

Minimum Requirements:

- One human player competes against a computer player.
- Game is displayed using **ASCII characters** (+, -, |) in the terminal.
- Score is updated in real-time.
- Computer player must make valid moves autonomously.

Extra Credit Opportunities (5 pts each):

- Simulate **register behavior** using global/local variables and structures.
- Add a **graphical interface** using `ncurses`.(A C language library)
- Use **file I/O** to save and reload game state.
- Implement a **strategy AI** for the computer player.

Submission Requirements:

Submit a **ZIP** file containing:

1. **Report** (Word or PDF):

    a) Explanation of how the game works and architectural principles you applied.
    b) Challenges faced and solutions.
    c) Insights gained about computer architecture through coding.
    d) Description of any low-level simulation or optimizations performed.
    e) Suggestions for future improvements (optional).

2. **Demonstration Video** (or a link to YouTube), showing the game in action with audio explanation.
3. **Source Code Files**:
    - All `.c` or `.cpp` files and makefile/build instructions.
    - Modular structure with proper commenting is expected.
4. **User Manual**:
    - Instructions for compiling and running the game.
    - Note platform used (Linux/Windows recommended with standard compilers like `gcc`, `g++`, or Visual Studio).

Grading Rubric:

- **Correctness and Reflection of Architectural Principles**
- **Documentation and Demonstration**
