CMPE-220 Sec 01

Program Layout & Execution Report

SUBMITTED BY TEAM 09

Shreya Pudukkottai (017416724)

Bhavika Sodagum (017506567)

Parth Gala (018198661)

Om Bharatbhai Kapadiya (018282355)

Date: Dec 5, 2025

GitHub Repository Link

Project GitHub Repository

All project code - C recursion example, assembler, software CPU, and assembly program is available in the GitHub Repo.

GitHub URL:

https://github.com/shreyapbk0622/CMPE220_Assignment2

The repository contains:

- src/ folder with all source files.
- A README.md file with compile and run instructions.
- A report/ folder containing this report.
- A demonstration video file showing function calls and recursion on the software CPU.

How to Download, Compile, and Run

How to Download the Code:

1. Open the GitHub repository link in a browser.
2. Click on Code → Download ZIP and extract it, or clone it using:
3. git clone https://github.com/shreyapbk0622/CMPE220_Assignment2.git
4. Navigate into the project directory: cd cmpe220-recursion-project

How to Compile and Run the C Recursion Example

1. Change into the src directory: cd src
2. Compile the recursive factorial program:

   gcc -std=c11 factorial.c -o factorial

3. Run the program:

   ./factorial

This program defines a recursive factorial(int n) function and a main() driver that calls it and prints the result.

How to Compile and Run the Software CPU + Assembler Demo

1. Still inside the src directory, compile the assembler:

   gcc -std=c11 assembler.c -o assembler

2. Assemble the recursive program written in the custom ISA:

```
./assembler call_demo.asm call_demo.h
```

This generates call_demo.h and call_demo.bin. The header file contains the machine code array that is included by cpu.c.

3. Compile the software CPU:

```
gcc -std=c11 cpu.c -o cpu
```

4. Run the software CPU:

```
./cpu
```

The CPU loads the machine code from call_demo.h into memory and executes it.

The console output shows:

- The instruction pointer (IP).
- The stack pointer (SP).
- The current instruction (including CALL and RET).
- Register values after each instruction.
- The final recursion depth, demonstrating nested function calls and returns.

The behavior observed here is used in the video to explain how function calls and recursion are implemented using the custom software CPU design.

Team Member Contributions

1. **Shreya Pudukkottai**

- Implemented the recursive C function (factorial.c) and the main driver program.
- Integrated the previous Software CPU project (cpu.c) with the new assembler workflow.
- Developed and tested the custom assembly program (call_demo.asm) demonstrating recursion using CALL/RET.
- Debugged CPU behavior, instruction execution, stack usage, and recursion depth.
- Recorded the demonstration video explaining function calls, stack behavior, and recursion.
- Organized the GitHub repository and created the project README.

2. **Bhavika Sodagum**

- Assisted in analyzing the ISA requirements for recursion and verifying CALL/RET behavior.
- Helped structure the memory layout explanation and ensured consistency with the earlier CPU design project.
- Contributed comments and documentation inside assembler.c and cpu.c.

3. Parth Gala

- Helped test the assembler by assembling various .asm files and validating correct machine code output.
- Verified stack pointer transitions (SP-- / SP++) during CALL and RET to confirm correct recursion handling.
- Assisted with report preparation and diagram layout (memory map, instruction flow).

4. Om Kapadiya

- Contributed to debugging the CPU execution trace and verifying correct IP (Instruction Pointer) flow.
- Helped refine the recursion demo scenario and validated that the base case and recursive case behaved correctly.
- Assisted in preparing the video outline and ensuring all required demo steps were included.