

# Function Calls and Recursion on SC8 CPU

## CMPE220 - Systems Software Project - 2 - Program Layout & Execution

## Function Calls and Recursion Implementation on SC8 CPU

### Course:

CMPE220 - Systems Software

### Semester:

Fall 2025

### Team Members:

- Neel Asheshbhai Shah
- Vedant Tushar Mehta
- Aarav Pranav Shah
- Harshavardhan Kuruvella

## Video Demonstration

See [Video](#) for video.

# GitHub Repository

## Repository URL:

- [https://github.com/SpartaNeel1010/Program\\_layout\\_execution](https://github.com/SpartaNeel1010/Program_layout_execution)

## Repository Contents:

- Complete C source code with recursion examples
- SC8 assembly implementations
- Comprehensive documentation
- Video demonstration
- Build scripts and Makefile
- Project report

# How Everything Works Together

## Complete Workflow:

### 1. **Write Assembly Code** (factorial.asm, multiply.asm)

- Use SC8 instruction set
- Implement recursive logic
- Add I/O for output

### 2. **Assemble to Binary**

```
make asm-programs
```

- Lexer tokenizes the code
- Parser validates syntax
- Symbol table resolves labels
- Binary file generated

### 3. **Execute on Emulator**

```
make run-asm
```

- Emulator loads binary into memory
- CPU fetches, decodes, executes instructions
- Stack manages recursion
- Results displayed via I/O

### 4. **Debug if Needed**

```
make debug-factorial
```

- Step through each instruction
- Watch registers and stack
- Understand execution flow

# How to Download, Compile, and Run

## Prerequisites

Before running this project, you need:

1. **GCC Compiler** (for C programs)
  - Linux: `sudo apt install gcc`
  - macOS: `xcode-select --install`
  - Windows: Install MinGW or WSL
2. **G++ Compiler** (for SC8 CPU)
  - Usually comes with GCC
  - Linux: `sudo apt install g++`
  - macOS: Included with Xcode Command Line Tools
3. **Make Utility** (for build automation)
  - Usually comes with GCC
  - Linux: `sudo apt install make`
  - macOS: Included with Xcode Command Line Tools

**Note:** The SC8 CPU is included in this project under `CPU(project 1)/` and will be built automatically.

## Step 1: Download the Project

### Option A: Clone from GitHub

```
git clone https://github.com/SpartaNeel1010/Program_layout_execution
cd Program_layout_execution
```

## Step 2: Build Everything

**Build CPU, C programs, and assemble programs:**

```
make all
```

This will:

1. Build the SC8 CPU (assembler and emulator)
2. Compile both C programs (factorial and multiply)
3. Assemble both SC8 assembly programs

### **Build components separately:**

```
make cpu           # Build SC8 CPU only
make c-programs    # Build only C programs
make asm-programs  # Build only assembly programs
```

## **Step 3: Run the Programs**

### **Run Assembly Programs on SC8 Emulator:**

```
# Run all assembly programs
make run-asm

# Or run individually
make run-asm-factorial
make run-asm-multiply

# Or run directly
./CPU\project\ 1\)/bin/emulator assembly/factorial.bin
./CPU\project\ 1\)/bin/emulator assembly/multiply.bin
```

# Team Member Contributions

## Neel Asheshbhai Shah

**Role:** Project Lead, Assembly Implementation

**Contributions:**

- Implemented SC8 assembly versions (factorial.asm, multiply.asm)
- Developed function call and recursion mechanisms in assembly
- Wrote MEMORY\_LAYOUT.md documentation with execution traces
- Tested and debugged assembly code on SC8 emulator

## Vedant Tushar Mehta

**Role:** C Implementation, Documentation

**Contributions:**

- Implemented C versions of recursive functions (factorial.c, multiply.c)
- Wrote FUNCTION\_CALLS.md documentation with call mechanism diagrams
- Performed comparative analysis between C and assembly implementations
- Added comprehensive code comments and verified outputs

## Aarav Pranav Shah

**Role:** Documentation, Testing

**Contributions:**

- Wrote RECURSION\_EXPLAINED.md with detailed execution traces
- Created stack evolution diagrams and test cases
- Developed project [README.md](#) with usage instructions
- Performed extensive testing and documentation review

## Harshavardhan Kuruvella

**Role:** Video Production, Build System

**Contributions:**

- Created VIDEO\_SCRIPT.md and recorded demonstration video
- Produced visual animations and memory layout diagrams
- Developed Makefile for automated building and testing
- Prepared final project report and presentation materials

## **Team Collaboration**

- Weekly team meetings and collaborative debugging sessions
- Peer review of all documentation and code
- Integration testing and GitHub repository organization