

Software CPU Design Project

CMPE220 - Systems Software - Final Project

Team Members

Neel Asheshbhai Shah

Vedant Tushar Shah

Aarav Pranav Shah

Harshavardhan Kuruvella

Course: CMPE220 - Computer Architecture

Semester: Fall 2025

Submission Date: November 26, 2025

GitHub Repository

The complete project source code, documentation, and demo video are available on GitHub:

Repository URL:

Example: https://github.com/SpartaNeel1010/CMPE220_CPU_DESIGN

Repository Contents

The repository includes:

- Complete source code for CPU emulator (C++)
- Complete source code for assembler (C++)
- Sample assembly programs (Timer, Hello World, Fibonacci)
- Comprehensive documentation (ISA specification, architecture details)
- Build system (Makefile)
- Project report (this document)

Download, Compile, and Run Instructions

Prerequisites

Before building the project, ensure you have the following installed:

- **C++ Compiler:** g++ or clang++ with C++11 support
- **Make utility:** For building the project
- **Git:** For cloning the repository
- **Operating System:** Linux, macOS, or WSL (Windows Subsystem for Linux)

Step 1: Download the Project

Option A: Clone from GitHub

```
git clone [your-repository-url]  
cd CMPE220_project
```

Step 2: Verify Prerequisites

Check that you have the required tools:

```
# Check C++ compiler  
g++ --version  
  
# Check Make  
make --version  
  
# Check Git (optional, for cloning)  
git --version
```

If any tools are missing:

On Ubuntu/Debian:

```
sudo apt update  
sudo apt install build-essential git
```

On macOS:

```
# Install Xcode Command Line Tools  
xcode-select --install
```

On Windows:

- Install WSL (Windows Subsystem for Linux)
- Then follow Ubuntu instructions

Step 3: Build the Project

Navigate to the project directory and build everything:

```
# Build emulator, assembler, and assemble all programs  
make all
```

This command will:

1. Compile the CPU emulator from C++ source
2. Compile the assembler from C++ source
3. Assemble all .asm programs to .bin format

Step 4: Run the Programs

After successful build, run the sample programs:

Run Hello World

```
make run-hello
```

Run Fibonacci Sequence

```
make run-fib
```

Expected output:

```
Running Fibonacci program...
=====
...
Fib: 0 1 1 2 3 5 8 ## ## ## Done!
...
```

Note: ## symbols represent numbers ≥ 10 (our simplified output only shows single digits)

Run Timer Example

```
make run-timer
```

Expected output:

```
Running Timer program...
=====
...
1
2
3
4
5
Done
...
```

Step 5: Run All Tests

To run all programs as a test suite:

```
make test
```

This will execute all three programs and display their output.

Custom Programs

To create and run your own assembly program:

1. Create a file: `programs/my_program.asm`
2. Assemble it: `./bin/assembler programs/my_program.asm programs/my_program.bin`
3. Run it: `./bin/emulator programs/my_program.bin`

Step 6: Clean Build Artifacts

To remove all compiled files:

```
make clean
```

This removes:

- Compiled binaries (bin/ directory)
- Assembled programs (.bin files)

Getting Help

For detailed documentation:

- ISA Reference: docs/ISA_SPECIFICATION.md
- Architecture Details: docs/CPU_ARCHITECTURE.md
- Main README: README.md

Team Member Contributions

Neel Asheshbhai Shah

CPU Emulator & Memory System

- Implemented CPU core with register file, program counter, and fetch-decode-execute cycle
- Developed instruction execution logic for arithmetic, logical, and branching operations
- Created 64KB memory subsystem with memory-mapped I/O (console output, timer)
- Conducted testing and validation of emulator functionality

Key Files: `src/emulator/cpu.{h, cpp}` , `memory.{h, cpp}` , `main.cpp`

Vedant Tushar Shah

Assembler & Build System

- Designed and implemented two-pass assembler with lexer, parser, and code generator
- Developed symbol table for label resolution and forward references
- Created instruction encoding scheme with multiple numeric format support
- Built comprehensive Makefile with build dependencies and test targets

Key Files: `src/assembler/assembler.{h, cpp}` , `lexer.{h, cpp}` , `parser.{h, cpp}` ,
`symbol_table.{h, cpp}` , `Makefile`

Aarav Pranav Shah

ISA Design & ALU Implementation

- Designed complete instruction set architecture with five instruction formats
- Implemented ALU with all operations and flag update logic (N, Z, C, V)
- Developed system bus simulation and control unit
- Created comprehensive ISA specification and architecture documentation

Key Files: `src/emulator/alu.{h, cpp}` , `bus.{h, cpp}` , `docs/ISA_SPECIFICATION.md` ,
`CPU_ARCHITECTURE.md`

Harshavardhan Kuruvella

Sample Programs & Documentation

- Developed three sample programs: Hello World, Fibonacci, and Timer with detailed comments
- Created comprehensive project documentation including README and demo script
- Wrote project report with all required sections
- Prepared demo video materials and presentation aids

Key Files: programs/*.asm , README.md , docs/DEMO_SCRIPT.md , report/PROJECT_REPORT.md

Team Collaboration

All members participated in design meetings, code reviews, testing, documentation review, and integration. Contributions tracked through Git with meaningful commit messages.