

Documentation

Date: 06/02/2026

Introduction

This project implements a simple calculator for an embedded system using QEMU

The software is divided into three logical layers, separating OS access, library-level C code, and user application logic

OS / Hardware Layer

- Purpose:

This layer provides direct interaction with the hardware, this code is provided in the laboratory instructions

- functions:

`uart_putc`: Function to send a single character via UART

`uart_getc`: Function to receive a single character via UART

`uart_puts`: Function to send a string via UART

`uart_gets_input`: Function to receive a line of input via UART

Notes:

This layer does not process numbers, formats, or arithmetic operations. It only handles raw data transfer.

Language Library Layer

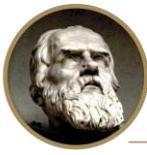
- Purpose:

This layer provides formatted input and output functionality similar to standard C libraries.

- Main functions:

`PRINT`: Formats and prints integers, fixed-point floats, and strings using UART output

`READ`: Reads user input and converts it into integers, fixed-point floats, or strings.



`print_int`: Converts an integer value to a string and prints it.

`print_float`: Prints a fixed-point number with two decimal places.

`uart_itoa`: Converts an integer into its string representation.

`uart_atoi`: Converts a numeric string into an integer value.

`int_div`: Simulates integer division using repeated subtraction

`float_div`: Wrapper function that performs fixed-point division using `int_div`

`have_decimal`: Checks whether a numeric string contains a decimal point

`float_from_string`: Converts a decimal string (e.g. 10.20) into a fixed-point representation (1020)

`parse_float`: Wrapper function that converts a numeric string into a fixed-point integer.

`parse_int`: Wrapper function that converts a numeric string into an integer

- Responsibilities:

Interpreting format specifiers (%d, %f, %s)

Converting integers and fixed-point numbers into strings

Formatting output for UART display

Logic to float numbers

User logic Layer:

- Purpose:

This layer contains the calculator logic and user interaction flow.

- Main:

Menu handling

Arithmetic operations (addition, subtraction, multiplication, division)

Fixed-point parsing and arithmetic functions

Control flow using while and switch

- Responsibilities:

Requesting user input

Making decisions based on user selection
Calling PRINT and READ for input/output

Summary

By separating the system into layers, the project achieves better modularity and clarity.

The OS layer manages hardware communication, the language library handles formatted input/output, and the user layer implements the calculator logic.

This design simplifies debugging and makes the system easier to extend or modify.