Lab 2 report

CSE121

Inhle Cele

Contents

Lab2.1: print value with GDB

Lab 2.1 findings

Compute 1st argument is 33 and is passed via register a0 → a4

Compute 2nd argument is 0 and is passed via register a1 → a5

Compute 3rd argument is 4 and is passed via register a2 → a5

Return value is 37 and is returned via register a0.

The entry point of the compute function is at address 0x4200bb7e

Steps:

1. **Starting GDB and Setting Up the Environment:** I used idf.py openocd to enable OpenOCD, allowing GDB to communicate with the ESP32 board. After that, I started GDB with the following command:

```
riscv32-esp-elf-gdb -x gdbinit build/lab2_debug.elf
```

   This initialized GDB and loaded the executable into the debugger.

2. **Locating the compute Function's Entry Point:** Using the riscv32-esp-elf-objdump command, I disassembled the lab2_debug.elf file and searched for the compute function's entry point. I used the following command to extract the symbol table and find the address:

```
riscv32-esp-elf-objdump -t build/lab2_debug.elf | grep compute
```

   This gave me the entry point address, which is 0x4200bb7e.

3. **Setting Breakpoints and Running the Program:** I set a breakpoint at the entry point of the compute function using:

```
b *0x4200bb7e
```

   After continuing execution (c command), the program halted at the compute function.

4. **Inspecting Registers for Function Arguments:** According to the RISC-V calling convention, function arguments are passed in registers a0, a1, and a2. Using the info registers command in GDB, I inspected these registers at the point where the compute function was called.

- **1st argument (33)** was passed via register a0 → a4.
- **2nd argument (0)** was passed via register a1 → a5.
- **3rd argument (4)** was passed via register a2 → a5.

5. **Finding the Return Value:** After the function completed execution, I examined the a0 register again, as the return value is stored there. The return value was **37**, and it was found in register a0.
6. **Locating the Return Address:** Finally, I identified the return instruction by stepping through the function using ni and looking for the instruction that transfers control back to the caller. The return address for the compute function was found at 0x4200bb7e

ChatGPT chat export:

User: help me do this: Lab 2 - Debugging and Humidity and Temperature

Due Date: Wed 10/23/2024

This lab is worth 20 Points. Project check-off will take place some time after the lab has been submitted to Gradescope during a future scheduled TA section. The overall objective of this lab is to use GDB and interface with a RISC-V executable, and to implement a humidity/temperature sensor.

It is VERY important to submit the report.pdf file. If this file is missing, you lose 1/2 of the points.

Lab2.1: print value with GDB (10 points)

This repo has a "lab2/lab2_debug.elf" file. You should convert it to a .bin, then upload it to the ESP32 board, then run GDB with it.

Some things that you will need to "figure out".

How can I upload an image obtained from someone else (where I do not have the source code)? You will not be able to compile it from the source. There are many ways to do this.

How can I use gdb against a binary/image

The code was compiled without debug info, so you have to look at the assembly code. If you forgot the RISC-V function call convention, the following document explains it: https://riscv.org/wp-content/uploads/2015/01/riscv-calling.pdf

One word of caution. Beware with how of the use of idf.py flash since this command not only tries to

Lab2.2: humidity and temperature

Lab 2 was much easier, and the code worked as expected, I just gave chat the esp32 specifications and it helped me write the code. Tutor checked me off and the gradescope autograder compiled without errors.

ChatGPT chat export:

Chat Export

The conversation includes the following:

- User's request for help with ESP32 programming and sensor interfacing using I2C.
- Discussion about working with the SHTC3 temperature and humidity sensor on an ESP32C3.
- Issues around getting constant temperature and humidity updates.
- Troubleshooting the ESP32 green light.
- Full code implementation using ESP-IDF?s i2c.h library.