2.1

Compute 1st argument is 33 and is passed via register a0.
Compute 2nd argument is 0 and is passed via register a1.
Compute 3rd argument is 4 and is passed via register a2.
Return value is 37 and is returned via register a0.

The original values are a0 = 0x21 (33), a1 = 0x0, and a2 = 0x4 (4).
After adding a0 and a1, a4 becomes 0x21 (33).
Adding a4 to a2 gives the final result a5 = 0x25 (37).
The value a5 = 0x25 is moved to a0.

The entry point of the compute function is 0x4200bb7e.

I ran into a problem where c wouldn't work in gdb after setting the compute breakpoint. The fix was running "mon reset halt" every time gdb was started.

I executed the following commands:

```
cd ~/esp/esp-idf
./install.sh esp32c3
. ./export.sh
cp -a examples/get-started/hello_world ~/esp/
cd ~/esp/
mv hello_world lab2_1
cd esp/lab2_1/main
vim hello_world_main.c
cd ..
idf.py set-target esp32c3
idf.py flash monitor
cd build
rm hello_world.bin hello_world.elf
cp ~/Downloads/lab2_image.elf ~/esp/lab2_1/build/hello_world.elf
esptool.py --chip esp32c3 elf2image hello_world.elf
esptool.py --chip esp32c3 write_flash 0x10000 hello_world.bin
lsusb
sudo chmod 666 /dev/bus/usb/001/005
idf.py openocd
source ~/esp/esp-idf/export.sh
vim gdbinit
```

```
riscv32-esp-elf-gdb -x gdbinit build/hello_world.elf
ubuntu@ubuntu:~/esp/lab2_1$ riscv32-esp-elf-gdb -x gdbinit lab2_image.elf
GNU gdb (esp-gdb) 14.2_20240403
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=aarch64-linux-gnu --target=riscv32-esp-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab2_image.elf...
warning: multi-threaded target stopped without sending a thread-id, using first
non-exited thread
0x4038e9f2 in wdt_hal_write_protect_enable ()
JTAG tap: esp32c3.tap0 tap/device found: 0x00005c25 (mfg: 0x612 (Espressif
Systems), part: 0x0005, ver: 0x0)
[esp32c3] Reset cause (3) - (Software core reset)
Warning: 'flushregs', an alias for the command 'maintenance flush register-cache', is
deprecated.
Use 'maintenance flush register-cache'.

Breakpoint 1 at 0x4200bc0c
Note: automatically using hardware breakpoints for read-only addresses.
^Cmo
Thread 1 "IDLE" received signal SIGINT, Interrupt.
0x4038eb9e in wdt_hal_feed ()
(gdb) mon reset halt
JTAG tap: esp32c3.tap0 tap/device found: 0x00005c25 (mfg: 0x612 (Espressif
Systems), part: 0x0005, ver: 0x0)
[esp32c3] Reset cause (3) - (Software core reset)
(gdb) b * compute
Breakpoint 2 at 0x4200bb7e
(gdb) c
```

```
Continuing.
[Switching to Thread 1070178524]

Thread 2 "main" hit Breakpoint 2, 0x4200bb7e in compute ()
(gdb) i r
ra             0x4200bcea  0x4200bcea <app_main+322>
sp             0x3fc9a050  0x3fc9a050
gp             0x3fc95c00  0x3fc95c00 <esp_flash_gpspi_host+24>
tp             0x3fc905f0  0x3fc905f0
t0             0x4005890e  1074104590
t1             0x20000000  536870912
t2             0x0     0
fp             0x3fc9a080  0x3fc9a080
s1             0x3c0234cc  1006777548
a0             0x21   33
a1             0x0     0
a2             0x4     4
a3             0x0     0
a4             0x4     4
a5             0x0     0
a6             0x0     0
a7             0xa     10
s2             0x0     0
s3             0x0     0
s4             0x0     0
s5             0x0     0
s6             0x0     0
s7             0x0     0
s8             0x0     0
s9             0x0     0
s10            0x0     0
--Type <RET> for more, q to quit, c to continue without paging--
s11            0x0     0
t3             0x0     0
t4             0x0     0
t5             0x0     0
t6             0x0     0
pc             0x4200bb7e  0x4200bb7e <compute>
(gdb) x/i $pc
=> 0x4200bb7e <compute>:        addi    sp,sp,-32
```

```
(gdb) si
[esp32c3] Target halted, PC=0x4200BB80, debug_reason=00000004
0x4200bb80 in compute ()
(gdb) x/i $pc
=> 0x4200bb80 <compute+2>:     sw      s0,28(sp)
(gdb) si
[esp32c3] Target halted, PC=0x4200BB82, debug_reason=00000004
0x4200bb82 in compute ()
(gdb) x/i $pc
=> 0x4200bb82 <compute+4>:     addi    s0,sp,32
(gdb) si
[esp32c3] Target halted, PC=0x4200BB84, debug_reason=00000004
0x4200bb84 in compute ()
(gdb) x/i $pc
=> 0x4200bb84 <compute+6>:     sw      a0,-20(s0)
(gdb) si
[esp32c3] Target halted, PC=0x4200BB88, debug_reason=00000004
0x4200bb88 in compute ()
(gdb) x/i $pc
=> 0x4200bb88 <compute+10>:    sw      a1,-24(s0)
(gdb) si
[esp32c3] Target halted, PC=0x4200BB8C, debug_reason=00000004
0x4200bb8c in compute ()
(gdb) x/i $pc
=> 0x4200bb8c <compute+14>:    sw      a2,-28(s0)
(gdb) si
[esp32c3] Target halted, PC=0x4200BB90, debug_reason=00000004
0x4200bb90 in compute ()
(gdb) x/i $pc
=> 0x4200bb90 <compute+18>:    lw      a4,-20(s0)
(gdb) si
[esp32c3] Target halted, PC=0x4200BB94, debug_reason=00000004
0x4200bb94 in compute ()
(gdb) x/i $pc
=> 0x4200bb94 <compute+22>:    lw      a5,-24(s0)
(gdb) si
[esp32c3] Target halted, PC=0x4200BB98, debug_reason=00000004
0x4200bb98 in compute ()
(gdb) x/i $pc
=> 0x4200bb98 <compute+26>:    add     a4,a4,a5
```

```
(gdb) si
[esp32c3] Target halted, PC=0x4200BB9A, debug_reason=00000004
0x4200bb9a in compute ()
(gdb) x/i $pc
=> 0x4200bb9a <compute+28>:   lw       a5,-28(s0)
(gdb) si
[esp32c3] Target halted, PC=0x4200BB9E, debug_reason=00000004
0x4200bb9e in compute ()
(gdb) x/i $pc
=> 0x4200bb9e <compute+32>:   add      a5,a5,a4
(gdb) si
[esp32c3] Target halted, PC=0x4200BBA0, debug_reason=00000004
0x4200bba0 in compute ()
(gdb) x/i $pc
=> 0x4200bba0 <compute+34>:   mv       a0,a5
(gdb) i r
ra             0x4200bcea  0x4200bcea <app_main+322>
sp             0x3fc9a030  0x3fc9a030
gp             0x3fc95c00  0x3fc95c00 <esp_flash_gpspi_host+24>
tp             0x3fc905f0  0x3fc905f0
t0             0x4005890e  1074104590
t1             0x20000000  536870912
t2             0x0    0
fp             0x3fc9a050  0x3fc9a050
s1             0x3c0234cc  1006777548
a0             0x21   33
a1             0x0    0
a2             0x4    4
a3             0x0    0
a4             0x21   33
a5             0x25   37
a6             0x0    0
a7             0xa    10
s2             0x0    0
s3             0x0    0
s4             0x0    0
s5             0x0    0
s6             0x0    0
s7             0x0    0
s8             0x0    0
```

```
s9             0x0    0
s10            0x0    0
--Type <RET> for more, q to quit, c to continue without paging--
```

The code above shows each step of the compute function.

2.2

I completed part 2 on vscode on my mac.

I ran into a problem with the ESP32C3 where “ls /dev/tty.*” on my mac and “lsusb” on the raspberrypi were unable to detect tty.usbmodem1101 and ttyS0 respectively (these were the ports I was trying to connect my board to). After getting a new board from BELS three times that would work initially and then quickly be undetectable I found that the solution is to unplug, hold boot, and plug back in while holding boot, check for detection and then release the boot button.

I started a project on vscode with the i2c-tools template. I wrote main.c with the following ChatGPT prompts:

Prompt:
Write me a basic example with the temperature sensor working on my ESP32C3 board using IDF. RUST ESP32C3 Board Temperature and Humidity SHTC3 address: 0x70 I2C Bus Connection Signal GPIO SDA GPIO10 SCL GPIO8 Here is the info from the temperature datasheet: Each measurement cycle contains a set of four commands, each initiated by the i2c START condition and ended by the i2c STOP condition: 1. Wakeup command (0x3517) 2. Measurement command (0x7CA2) Enable clock stretching, read temperature 3. Read out command (just read) 4. Sleep command (0xB098)

Response:
Unused Code

Prompt:
Error messages

Response:
Unused Code

Prompt:

Error messages
Response:
Unused Code

Prompt:
Lab2.2: humidity and temperature (10 points) In a prior lecture we talked about the, SHTC3 temperature and humidity sensor contained in the ESP32C3 Rust board, and discussed how the I2C protocol for communicating with this device works. For this lab, you will have to write a program that reads the temperature and humidity, and prints the temperature in degrees C and F and a percentage for the humidity once every 2 seconds. The output will look something like this (round to nearest value): Temperature is 20C (or 68F) with a 40% humidity Temperature is 22C (or 72F) with a 42% humidity Temperature is 20C (or 68F) with a 44% humidity ... Some special requirements: You should use the power up and power down between reads (In class, we did not do power down). The power up should be called at most once every 2 seconds. You should have a different function for temperature and humidity. Each should read at most 3 bytes Use the checksum value to check the read results

Response:
Unused Code

Prompt:
Why am I getting vector tables (error output)

Response:
Unused Code

Prompt:
write the simplest code you can for this

Response:

Here’s the simplest version of your program that reads the temperature and humidity from the SHTC3 sensor, handles power up and down, and prints the values every 2 seconds. I've stripped it down to the essentials:

Simple Code for Reading Temperature and Humidity:

```
#include <stdio.h>
#include "driver/i2c.h"
#include "freertos/task.h"
```

```
// I2C configuration
#define I2C_MASTER_SCL_IO 8          // GPIO for SCL
#define I2C_MASTER_SDA_IO 10         // GPIO for SDA
#define I2C_MASTER_NUM I2C_NUM_0     // I2C port number for master
#define I2C_MASTER_FREQ_HZ 100000    // I2C clock frequency
#define SHTC3_SENSOR_ADDR 0x70       // SHTC3 I2C address

// SHTC3 command codes
#define SHTC3_CMD_WAKEUP 0x3517
#define SHTC3_CMD_SLEEP 0xB098
#define SHTC3_CMD_MEASURE 0x7CA2     // Measure temp and humidity with clock stretching

// Initialize I2C master
static void i2c_master_init(void) {
    i2c_config_t conf = {
        .mode = I2C_MODE_MASTER,
        .sda_io_num = I2C_MASTER_SDA_IO,
        .sda_pullup_en = GPIO_PULLUP_ENABLE,
        .scl_io_num = I2C_MASTER_SCL_IO,
        .scl_pullup_en = GPIO_PULLUP_ENABLE,
        .master.clk_speed = I2C_MASTER_FREQ_HZ,
    };
    i2c_param_config(I2C_MASTER_NUM, &conf);
    i2c_driver_install(I2C_MASTER_NUM, I2C_MODE_MASTER, 0, 0, 0);
}

// Send command to SHTC3 sensor
static esp_err_t shtc3_write_cmd(uint16_t cmd) {
    uint8_t cmd_buf[2] = { (cmd >> 8) & 0xFF, cmd & 0xFF };
    return i2c_master_write_to_device(I2C_MASTER_NUM, SHTC3_SENSOR_ADDR, cmd_buf, sizeof(cmd_buf), pdMS_TO_TICKS(1000));
}

// Read temperature and humidity data (2 bytes each)
static esp_err_t shtc3_read_data(uint8_t *data, size_t len) {
    return i2c_master_read_from_device(I2C_MASTER_NUM, SHTC3_SENSOR_ADDR, data, len, pdMS_TO_TICKS(1000));
}
```

```
// Convert raw sensor data to temperature (Celsius)
static float get_temperature(uint16_t raw_temp) {
    return -45 + 175 * (raw_temp / 65535.0);
}

// Convert raw sensor data to humidity (percentage)
static float get_humidity(uint16_t raw_hum) {
    return 100 * (raw_hum / 65535.0);
}

// Read and print temperature and humidity
static void read_and_print_data(void) {
    uint8_t data[6];

    // Power up the sensor
    shtc3_write_cmd(SHTC3_CMD_WAKEUP);
    vTaskDelay(pdMS_TO_TICKS(10));

    // Start measurement
    shtc3_write_cmd(SHTC3_CMD_MEASURE);
    vTaskDelay(pdMS_TO_TICKS(15));

    // Read 6 bytes: temperature (2 bytes), humidity (2 bytes), and 2 CRC bytes
    shtc3_read_data(data, 6);

    // Convert raw data to temperature and humidity
    uint16_t raw_temp = (data[0] << 8) | data[1];
    uint16_t raw_hum = (data[3] << 8) | data[4];
    float temperature = get_temperature(raw_temp);
    float humidity = get_humidity(raw_hum);

    // Print the results
    printf("Temperature: %.1f°C, Humidity: %.1f%%\n", temperature, humidity);

    // Power down the sensor
    shtc3_write_cmd(SHTC3_CMD_SLEEP);
}

void app_main(void) {
    i2c_master_init();

    while (1) {
        read_and_print_data();
        vTaskDelay(pdMS_TO_TICKS(2000));  // Wait for 2 seconds
    }
}
```

Key Points:

1. I2C Setup: The I2C master is initialized with GPIO10 (SDA) and GPIO8 (SCL) at 100 kHz.
2. Power Up/Down: The sensor is powered up before reading and powered down afterward.
3. Simple Temperature and Humidity Read: The program reads 6 bytes from the sensor (2 for temperature, 2 for humidity, and 2 CRCs) and converts the raw values into human-readable data.
4. Output: The temperature and humidity are printed every 2 seconds.

Running the Code:

- Ensure proper wiring for the I2C bus (SDA to GPIO10, SCL to GPIO8).
- Load the code onto your ESP32C3 and observe the temperature and humidity readings printed every 2 seconds.

Let me know if this works for you!

This was all the ChatGPT prompts used. I changed a line to convert the temperature to fahrenheit so the output is more similar to the lab example:
From:

```
printf("Temperature: %.1f°C, Humidity: %.1f%%\n", temperature, humidity);
```

To:

```
printf("Temperature is %.1f°C (or %.1f°F) with a %.1f%% Humidity\n", temperature, temperature * 1.8 + 32, humidity);
```