Report on Coursework 2 Systems Programming

Group no : 28

Name of pair : Fahad Mohammad Ali , Saad Mirza

Problem Specification:

We have been assigned with the responsibility of developing a rudimentary edition of the Mastermind game on a Raspberry Pi 3, which will involve the utilization of LEDs, a button, and an LCD. Our implementation will entail the use of the C programming language, along with some ARM assembler code that will be integrated into the C codebase to enhance our comprehension of the hardware's inner workings at a fundamental level.

Hardware Specification and Platform:

The equipment utilized for this academic assignment comprises the Raspberry Pi 2 microcomputer, a breadboard, an LCD display, two LEDs, and a button, all connected through appropriate wires and resistors. Every connection with any external input/output device is established by attaching the device to a specific RPI2 pin, which is subsequently mapped into memory to facilitate its usage by writing in the correct position in memory. In this project, every input/output device is linked to the breadboard, and the wire connected with the corresponding pin is also inserted in the appropriate column of the breadboard to transmit the data to the device. Each circuit terminates with a resistor and another wire that is connected to the ground. The two LEDs are linked to pin numbers 5 and 13, respectively, and configured as output, while the button is attached to pin number 19 and configured as input. As for the LCD, it necessitates a more intricate wiring approach since more data needs to be transmitted to it. Specifically, we connect it to four distinct GPIO pins to convey the bits, which are pins 23, 10, 27, and 22.

Code structure:

The code structure adheres to the conventional C structure, with include and define statements at the outset, followed by the functions necessary for the main function, with the main method constituting the final portion. Numerous functions are declared that are essential for the display's operation, while others, such as delay functions, serve to insert pauses in the program flow. I also crafted functions for the game's functionality, such as the one that identifies the number of exact and partial matches or the one that verifies if a value exists in an array. Additionally, the program encompasses three functions to interact with the hardware, which will be further elaborated on in the ensuing paragraph. The main function commences with the declaration of variables necessary for program execution, followed by memory mapping to transfer the GPIO pins into main memory. This allows us to send bits by writing in the correct location in memory. After mapping the pins in memory, we proceed to set the modes for the IO devices connected to those pins. We then configure the display mode and ensure that it is ready to receive characters via the lcdPuts() function. At this juncture, we commence game execution, which will be discussed in the "program execution" paragraph.

Functions accessing hardware:

In my program I have three functions to communicate directly to the hardware:

- void ledpinMode(uint32_t *gpio, int pin, int mod
- void writeLED(uint32_t *gpio, int pin, int value)
- readButton(uint32_t *gpio, int button)
- void waitForButton(uint32_t *gpio, int button)

ledpinMode: This function is responsible for setting the mode of a GPIO pin and takes in three arguments: a pointer to a 32-bit integer representing the GPIO pin, an integer pin number, and an integer mode. The function employs assembly language to manipulate the binary values of the register that manages the mode of the given pin. Initially, it calculates the offset of the pin number and shifts it to the appropriate position. Next, it loads the existing register value, clears the bits that regulate the pin's mode, sets them to the desired mode, and writes the updated value back to the register. This function is utilized to set the mode of several components such as the button, two LEDs, and the four pins connected to the display.

WriteLED: This function takes three parameters: a pointer to a 32-bit integer representing the GPIO pin, an integer pin number, and an integer value. It uses assembly language to write to the binary values of the register that controls the pin's output state. If the value is 1, the function sets the specified pin to 1, and if the value is 0, it sets the specified pin to 0. The function uses conditional statements to check whether the pin number is 5 or 13 and sets the corresponding pin to the specified value. This function is used to write to the two LEDs connected to pins 5 and 13.

readButton: This function reads the value of a GPIO input pin connected to a button. It takes two parameters, a pointer to a 32-bit integer representing the GPIO pin and an integer button number. The function calculates the offset for reading the input and the pin to be read. It then loads the current register value, performs a bitwise AND operation with the desired pin to get the value of that pin, and stores the result in the return variable. If the value is not 0, the function returns 1, otherwise, it returns 0. The function uses assembly language to perform these operations efficiently.

waitForButton: This function waits for a button press. It takes two parameters: a pointer to a 32-bit integer representing the GPIO pin and an integer button number. The function uses a while loop that checks the value of the button using the readButton() function until the button is pressed. Once the button is pressed, the function exits the loop, and the program can continue executing.

Program Execution:

When the program is executed in debug mode (by adding the argument "-d" when running it), the following sequence is followed:

1. A message "Press Enter to continue" is displayed on the terminal, and the program waits for the user to press enter to start the game.
2. The welcome message is printed on the LCD.
3. The red LED blinks twice to indicate that the game has started.
4. The secret code is generated and displayed on the terminal.
5. The first round starts, where the user is required to input the code using the button. The LEDs blink once for every button press.
6. Red LED blinks once to indicate that the first code of the secret has been guessed.
7. For every code input (3 inputs in total), the exact and partially correct guesses are displayed on the LCD.
8. The game finishes if the secret code is guessed (the number of attempts is displayed on the LCD) or if the player has reached the maximum number of attempts, which is set to 3.

Each Members contribution in this project:

In this pair Mastermind project, both members made significant contributions to its success. The first member focused on implementing the game's functionality, which included generating the secret code, receiving, and validating the player's guesses, and providing feedback on the correctness of each guess. This member also worked on the LCD display and the LED blinking to provide visual cues to the player. Meanwhile, the second member concentrated on the assembly part of the project, which was the most challenging aspect of the project. This member worked on creating functions to interact with the GPIO pins, ensuring they functioned correctly and efficiently. This included setting the pin mode, sending a 1 to either the clear or set register, and reading values from the GPLEV register. Overall, both members collaborated effectively, communicated well, and provided feedback to each other to produce a quality final product.

Summary:

Throughout the completion of this coursework, I have gained extensive knowledge about low-level programming, particularly in the assembly language aspect, which I found to be the most challenging. This experience has taught me the value of a hardware-oriented approach. I am confident that I have successfully accomplished all the required features outlined in the coursework specification, including the game functionality and inline assembler implementation. Initially, I created separate functions for each task involved in the assembler part. Consequently, I created these three functions. Overall, I thoroughly enjoyed working on this coursework, and I believe it has significantly improved my programming skills.