F28HSI Hardware-Software System

Coursework-2 : Report

Members:

1) Nidal Arafath – mna2002@hw.ac.uk
2) Cheryl Mattam – cbm2001@hw.ac.uk

I. Problem Specification

The aim of the coursework is to develop a simple, systems-level application in C and ARM Assembler, running on the Raspberry Pi with attached devices.

The application, Mastermind is a two-player game between a codekeeper and codebreaker. The codekeeper chooses a pattern of N code pegs and places them into a sequence of N slots, visible to the codekeeper but not to the codebreaker. In rounds, the codebreaker tries to guess the hidden sequence and, in each round, the codekeeper answers by stating by how many pegs are of the right color and at the right position and how many pegs are of the right color but not at the right position. The game is over when the codebreaker successfully guesses the code, or if a fixed number of rounds has been reached.

II. Hardware Specification and Wiring

For the hardware platform, the application needs to run on a Raspberry Pi 2 with the following attached devices : two LEDs and a button and resistors for all the three and all the devices should be connected to the RPi2 via a breadboard.

Wiring : The two LEDs are used as output: green LED is used for data which is connected to the RPi2 using GPIO pin 13 and red LED is used for controlling information which is connected to the RPi2 using GPIO pin 5. There are resistors connected to the negative leg of the LEDs which from there goes to the ground. For the button used as input, the first pin is connected to 3.3v , the second pin is connected to GPIO pin 19, and the last pin is connected to a resistor which goes to the ground.

III. Code Structure

A short discussion of the code structure, specifying the functionality of the main functions. The code structure is as follows:

initSeq: This function initializes a secret random sequence with length of "seqlen". rand() function is used to achieve this.

showSeq: Function to print the random secret sequence initialized by the program. This is utilized when the program is run in debug mode.

countMatches: Function to calculate the exact and approximate matches of the 2 sequences that is passed in the parameter. The exact and approximate matches are returned as a pointer array of length 2.

showMatches: This function calls the countMatches function and then prints the exact and approximate matches in the terminal.

readSeq: Function to convert an integer of digits to a pointer array of digits.

readNum: This function receives the guess sequence input from the user from the terminal. This is an alternative for the button presses and is utilized while testing.

timerInMicroseconds: This function returns the current time in microseconds.

time_handler: This is the function that is called when the timer expires. It can be used to indicate the end of the timer in the main program.

initTimer: Function that initializes timer based on the argument provided. This is a non-recurring timer i.e, it expires after the defined timeout.

blinkN: This function will blink the provided led defined number of times. The led and the number of times it must be blinked must be given as the argument. It turns the led on and off with a certain specified delay.

main: This is the main program that runs when the code is run. It holds the entire structure of the master mind game. It also includes the code for different modes/flags that the user can user can run the program in.

IV. Design Choices

A discussion of performance-relevant design decisions, and implications on resource consumption.

- Timer

A 5 second timer is used to get the input from the user.

The timer is only initialized when the user presses the button for the first time while inputting one number.

This way successive inputs can be read from the user improving efficiency and accuracy.

- Use of pointers

Pointer array is used to store the exact and approximate matches throughout the program.

The pointer is malloced at runtime with the size of 2 to hold each of the matches.

No extra ceremonies are required to access each of the values separately as indexing can be used this way.

- Assembly

Inline assembly is used in lcdBinary file to access the hardware such as the led and the button.

Use of assembly increases the accuracy since the registers are directly controlled by the code and this also increases the speed of the program.

V. Function Descriptions

A list of functions directly accessing the hardware (for LEDs, Button, and LCD display) and which parts of the function use assembler and which use C

The functions to access the hardware is in the lcdBinary.c file. It has the following structure:

pinMode: This function sets the gpio pin to input or output mode based on the argument passed. The part to determine the fSel and shift bit of the pin is implemented in C using a switch case. If an undefined pin is passed in the function, the program fails.

The mode of the pin is set using inline assembly

writeLED: This function is used to turn the defined led on and off. It writes the specified mode to the set or clear register. C is used to determine whether to write the value to the set or clear register. Inline assembly modifies the set/clear register to turn the led on and off.

readButton: Function to read button click by the user. C is used to determine the gplev0 register based on the pin the button is connected to. Inline assembly reads from the determined register and returns the value.

waitForButton: Function that waits until the user presses the button. It makes use of the readButton function.

VI. Matching Function Description

Discussion on the matching function implemented in ARM Assembly

The matching function, countMatches is implemented as inline assembly in the mastermind.c file. The two parameters passed as input to the file is the secret sequence and the guess sequence. Within the function, another sequence is defined in which the exact and approximate number of matches are stored. There is also a variable used as the approximate indicator The structure of the sub-routines defined is as follows

main_loop : This is the main part of the program that runs when the function is called. It takes in as input the secret and guess sequence and compares the values. Once the loop is done, it inputs the exact and approximate number of matches into a sequence and returns it.

loop_increment1: This subroutine is called in the main_loop to increment the variable used in the loop for traversal and it also increments the index variable used to access the elements of the secret and guess sequence.

add_exact: This subroutine takes in the guess sequence and the variable counting the approximate matches and compares them to ensure there are no duplicates and increments the variable counting the exact matches returning that to the main loop

check_approx_size: Subroutine to check if the element of the hidden sequence is already present in the approximate indicator variable and if it is , proceeds to the decrement_approx sub-routine.

decrement_approx : This subroutine decrements the approximate matches in the sequence to ensure there are no duplicates and returns the variable to check_approx_size sub-routine from which it was called.

approx_loop: This subroutine is called in the main_loop and traverses through the secret and guess sequences and checks if the elements of the guess sequence are present in the secret sequence but at different indexes.

loop_increment2: This subroutine is called in the approx_loop to increment the variable used in the loop for traversal and it also increments the index variable used to access the elements of the secret and guess sequence.

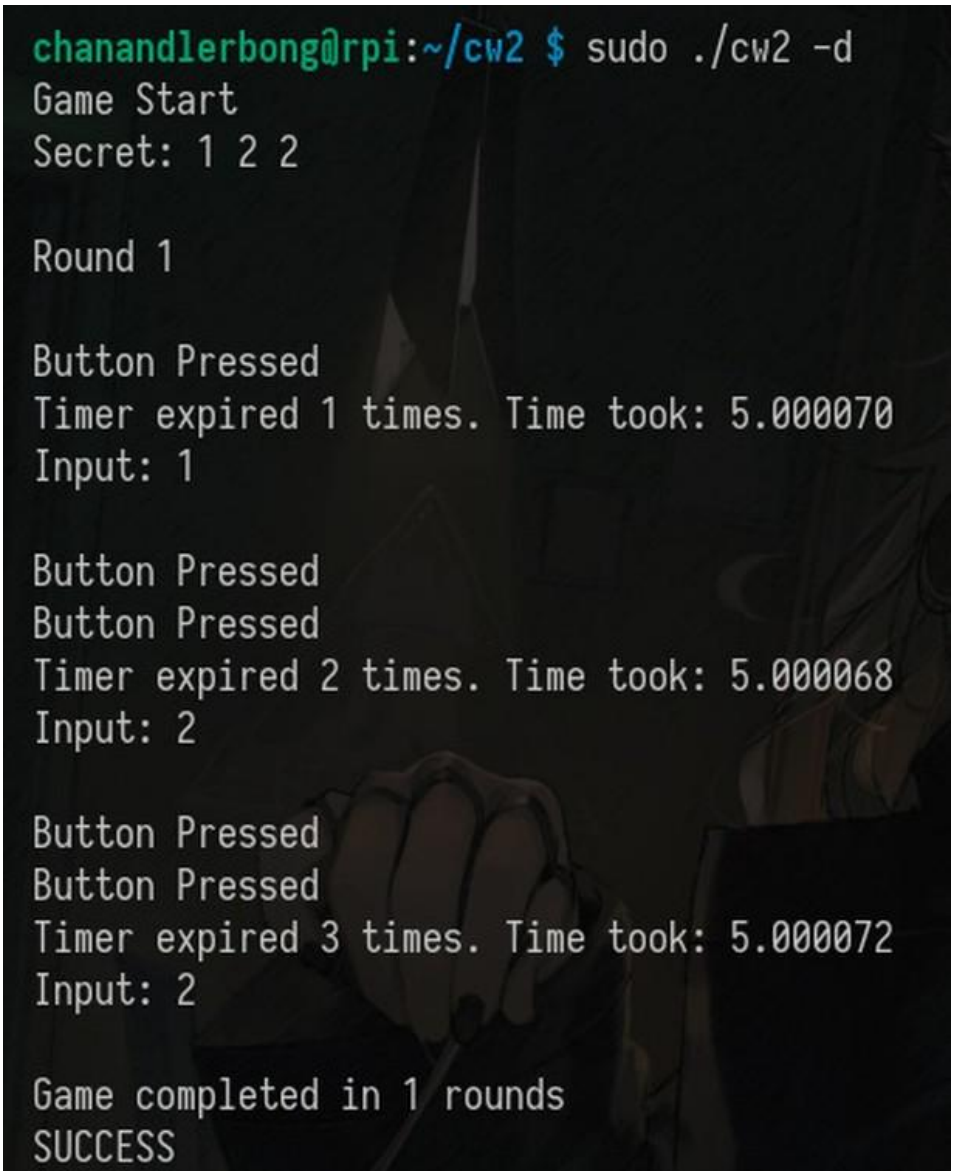
index_check: Subroutine to check if the indexes of the secret and guess sequence are not equal and if it's not, calls another sub-routine to make sure the element isn't repeated

approx_check: This subroutine checks if the element in the guess sequence isn't equal to the approximate indicator variable and if it is, it returns to approx._loop, else it proceeds to the next sub-routine.

add_approx : This sub-routine is called to increment the number of approximate matches and change the approximate indicator to the new approximate element of the guess sequence and returns to the main loop.

exit_routine: This sub-routine is called once we have compared all the elements of the secret and guess sequence. The two variables used to count the exact and approximate matches are now stored in a sequence.

VII. Execution of the program



VIII. Summary

All the functions implemented in the program played a role in successfully developing the Mastermind application.

Outstanding features:

- Our program meets all the specified requirements.

- It also implements assembly wherever specified making the code more efficient.
- No other extra features were added.

What we learned:

- This coursework taught us how to interact with the hardware which has been an exciting experience.
- Writing program to control physical objects was something fresh and enlightening.
- We learned how to write C with assembly together and produce an efficient code.
- We learned the to control the registers directly using assembly which was really thrilling.

Contributions by the members:

1) Nidal (mna2002@hw.ac.uk) :

the functions contributed to was

- countMatches (both arm and c)
- initSeq, readSeq, and showSeq
- timer
- main
- also contributed to the video demoing the running of the application and the final report

2) Cheryl (cbm2001@hw.ac.uk) :

the functions contributed to was

- blinkN
- readNum
- main
- implemented the lcdBinary file
- also contributed to the video demoing the running of the application and the final report