Group 16 (Mohamed Afifi, Jason Chen, Alex Rajanathan, Aliyan, Nicholas Theobald)

COM1031 Report

Part 1 – Overview of the implementation:

On the surface, the implementation starts with a welcome message, reads inputs through the timing of a press of a button, and ends by generating a morse code representation of one of twenty-six letters of the English alphabet. However, there are many steps and details required to fulfill such a sequence of tasks, best illustrated through a flowchart (figure 1).

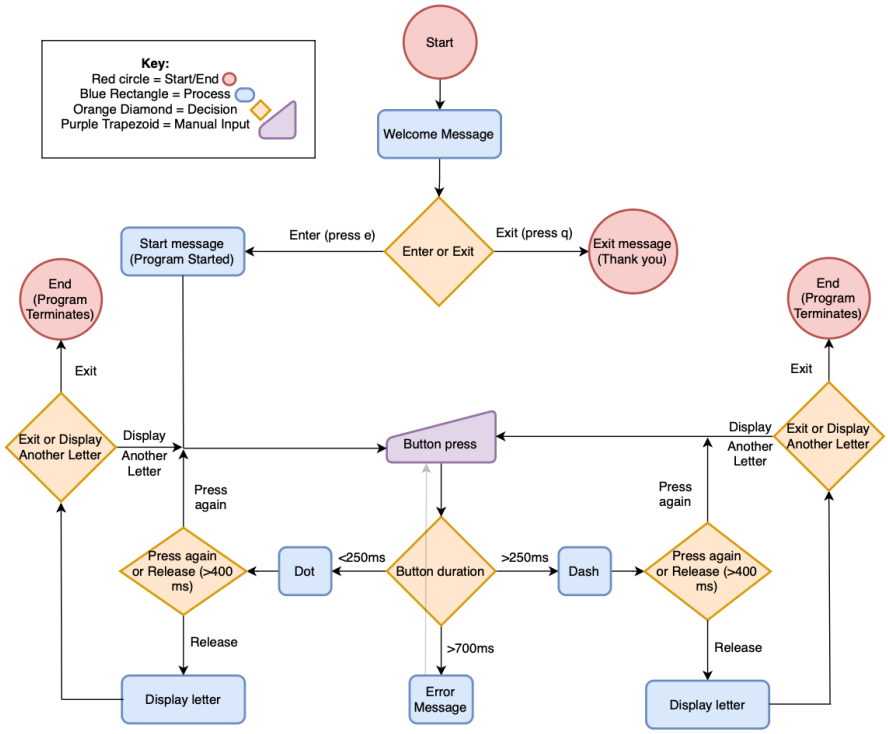

Figure 1 (diagrams.net)

Following the now displayed welcome message, the user is faced with a decision to enter or exit the morse code simulator by pressing e or q respectively. Whenever an exit path is chosen, it is assumed that the program always ends. Moreover, if the enter path is chosen, the user is presented with a welcome message as the program starts. Next, the program awaits the user's manual input (purple trapezoid) of a button press. Upon reading the given input, the program responds in one of three ways dictated by the user's decision. After being pressed, a button duration of less than 250 milliseconds generates a dot, while more than 250 milliseconds would generate a dash. However, if more than 700 milliseconds, an error message is displayed, and the user is stepped back to the button press stage. After a dot or dash is generated, the user faces a decision to either press the button again within 400 ms, in which the manual input and onwards steps are repeated, or to release the button, which in return, displays a letter within 400 ms. From this step, it is deduced that the release on the first cycle will only print one of two letters, an E (dot), or a T (dash). On the other hand, a "Press again" will represent two or more cycles, in which a combination of dots and dashes will print different letters accordingly. Finally, after every display of a letter, the user has the option to continue displaying more letters, or to terminate the program.

The discussed implementation was split into “Welcome function”, which included the start, welcome, and exit messages. In addition, a combination of many smaller programs and some additional assembly created what is known as loop reader. Everything was then put together into “morse”.

One of the key sections in loop reader is the read value function, where one of two paths are loaded (pin with one or two numbers), by reading the value from the pin. Next, the value of the string that was read is loaded, represented by the ASCII 48 and 49 for 0 and 1 respectively.

The identify function was responsible for setting up the parameters that would decide which letter will be printed out on the screen. Such a function followed a system where each letter from A to Z had a unique code starting with “2”, followed by a combination of “0” for dot and “1” for dash. The reasoning behind the “2” is because a number such as 001 is simply recognized as 1, which defeats the purpose. For example, “A” is represented by “201”, the 0 (dot) being the result of multiply by 10, and the 1 (dash) being the result of multiplying by 10, then adding 1.

Other functions that were utilized include the print dot and print dash functions, which set output as out, loads the pointer to the dot or dash string, and sets the function of the system call to write. In addition, dash add, and dot add print a string to the screen. Moreover, the error function of a button press longer than 700 ms exists to keep everything intact.

Part 2 – Challenges of the implementation:

A minor issue was present during the assembly of the program that displayed the letters A-Z on the seven-segment display, as the code was too long. The solution was a “.LTORG” instruction which gave the assembler the ability to collect and assemble literals into a pool. Another minor issue was being stuck in an infinite loop, which meant that when pressing the button for a dot or dash, it simply did not work. This was fixed by changing any b to a bl.

In addition, the loop reader section had a segmentation fault and could only guess morse codes with 4 inputs. This was fixed by popping and pushing registers to the stack, and utilizing “bx lr” branching instead of “b” branching. Misaligned branch destination errors were scattered throughout, and in most cases, were fixed by having multiple .data sections within the program.

Part 3 – Description of group contribution and member engagement:

The 76 marks of the criteria and 20 marks of the report were split between all five members to ensure the workload was evenly spread. However, as work progressed, different members collaborated on different criteria. The following list shows each member’s main contribution:

- **Mohamed Afifi:** Display three signal letters. Display four signal letters. >700 ms error function. Main report writer.
- **Jason Chen:** Connect external circuit. Display four signal letters. Review report.
- **Alex Rajanathan:** Input letters displayed correctly on seven segment display. Print dot function. Print dash function. Recognize and display error conditions.
- **Aliyan:** Welcome message. Seven segment display on/off. Seven segment display turns on when button is pressed. Display two signal letters. Program terminates with “q”. Input letters displayed correctly on seven segment display.
- **Nicholas Theobald:** Program detects button is pressed, displays character to terminal. Recognize and display one signal letters to terminal. Display two signal letters.