Computer Systems AC21009

Introduction

The aim of this project was to create a Manchester Baby simulator of the fetch-decode-execute cycle as well as an assembler program which assembles machine code.  To run the all programs, use the g++ compiler.

Design

The Design of the Manchester Baby was to be split up into two main classes, Mbaby and Menu. Mbaby would contain all functions needed for the operations of the program(such as conversions). Menu Would hold all the required fields(store,Current instruction, etc) and methods to run the machine. I did as much programming as I could for the project myself, implementing the menu and Manchester baby classes and all they included.

Due to a lack of communication from the entire group and organisation, I am unaware if the rest of the group has programmed anything themselves having reached out via message and email, not getting any response. Therefore, I cant speak for the rest of the group in terms of design.

Problems

Within the program, the main issue encountered was learning how the binary should be stored to allow for operations to be done to it.

One main issue was learning how to convert negative decimal to a binary, considering it required a different algorithm to a normal decimal.

The main problem with the Menu function is the main function infinitely loops due to some issue with the reading file where, the file isn’t read and stored correctly in the filestore for the program, so program doesn’t know when to end.

There was also a problem in testing the logic as due to either the read function or conversion function not working as expected, this made it impossible to test to see whether the code in execute was correct.

Solutions

After testing with storing the binary as a vector<int> and as an array, in the end it turned out the easiest and most effective way to store was to store them as string as the other 2 options were considerably harder to implement various functions with, such as the conversion ones.

A solution for the this problem was to first use the normal algorithm for positive decimals then use a overlay temporary binary string set to completely empty to flip it after to negative binary, iterating through the main binary string making the temp[index]=0 when main[index is]=1 and vice versa effectively flipping the string to negative. As the end of the process simply add a one to the end to return the correct binary for the negative value.

In order to solve the issue of the file not being read correctly(or converted correctly), the idea was to

Go back to check all the conversion and file methods and check to make sure 32 bit was being used. However due to time and personnel restrictions this was not possible.

Reflection

Looking back on the project, a lot went wrong and should’ve been rectified. The biggest issue was communication and organisation, myself being guilty for this. The group didn’t meet to discuss anything as far as I am aware, when I reached about the program/project, I only got one reply from Stephen Rae in the group late in the term, who had also said that no one had reached out to him and that he had not seen anyone in the group in labs or any messages/mail. Due to this, work was obviously not able to be split up for 5 people, meaning I had to program as much as I could myself. This meant that I essentially had no help in programming, so I ended up programming the Manchester Baby Simulator myself but due to time constraints and work load I was unable to begin implementing the assembler as well whilst also being unable to fully program the simulator program. To add, outside of Stephen and I, its entirely possible the other 3(who I didn’t hear from) did the project themselves without including us, meaning its possible that the group may end up with 2 different solutions/reports, making the solution/report I did myself redundant especially if I ended up coding something they already have. The program I implemented as a result of all this suffered and was an incomplete solution with the menu not running as expected.

Word Count:   707