Team Avengers 68K Disassembler Progress Report 3

Date: May 22nd, 2020

To: CSS 422, Spring 2020

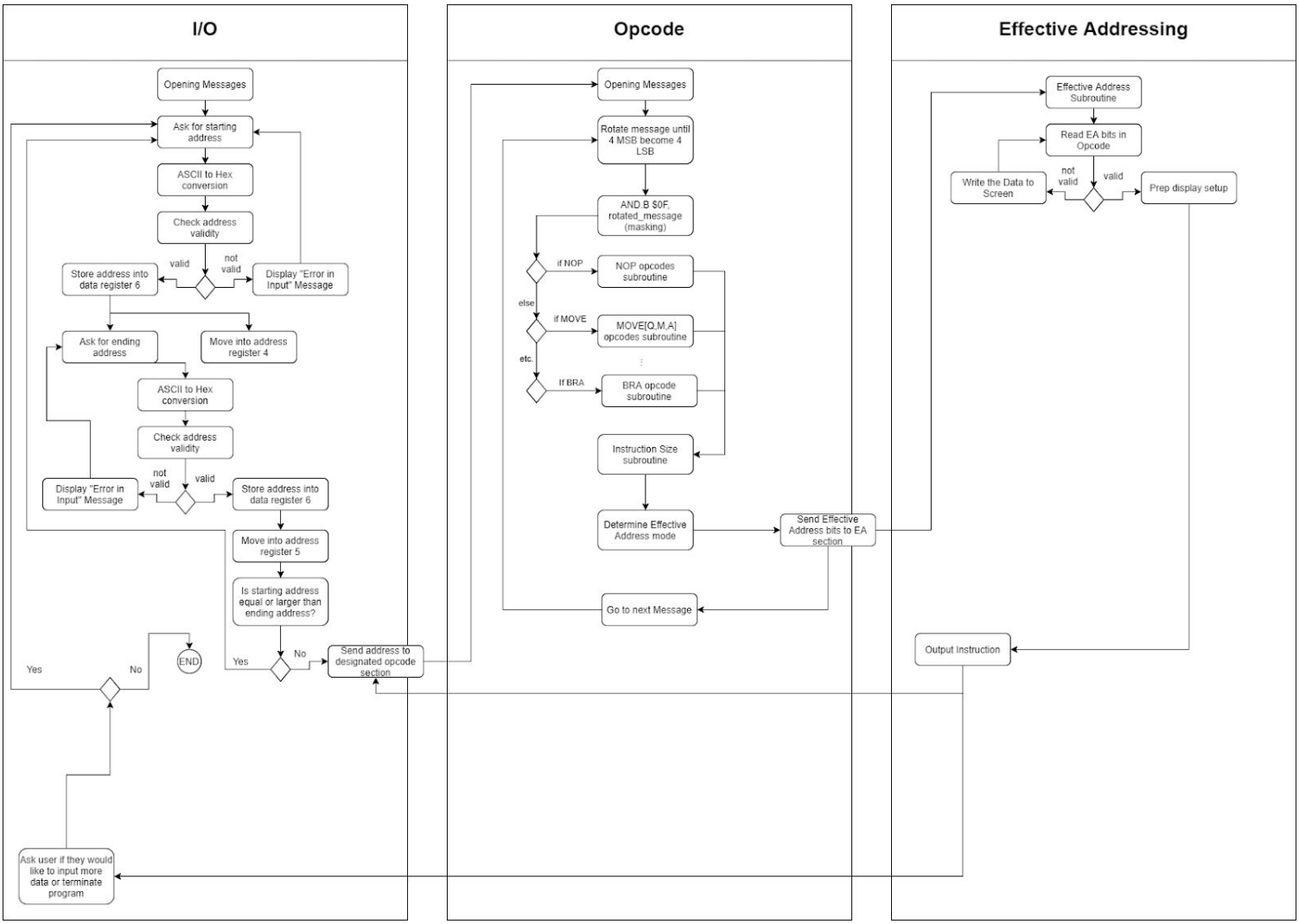
From: Group Avengers [Mohammed Ali, Dj Wadhwa, Tarcisius Hartanto]

Subject: 68k Disassembler Progress Report 3

Work Completed:

Mohammed completed the conversion for ASCII to hex for the I/O section. He also set up the user input section where the user inputs a 32-bit hex address. The input will then be converted to hex and stored into address registers. A memory search subroutine was also created that will go through each memory address between the inputted addresses. DJ has so far worked on implementing several subroutines for the easier opcodes including ADD, SUB, MOVE, MOVEA, and MOVEQ. We plan on implementing the MOVE.M as a group as it appears to be the most challenging instruction to read.

Tarcisius has been working with DJ to get the EA section started. He has also been working with Mohammed to get a Hex to ASCII section done. This will be used to output the memory addresses onto the screen along with the commands. We have decided to allocate each register for a specific purpose in each step. This has allowed everyone to understand which registers contain what kind of data at any given time. This has streamlined our coding process as currently, anyone is able to pick up the work from any section of the project without too much hassle.



List of Unfinished sections as of May 22nd, 2020:

1. **Completed Opcode Implementation**
    a. We have yet to fully complete the implementation of the opcode section. To get this part done we have agreed that DJ and Mohammed will work on the opcode section to finish it efficiently. Given the long weekend due to Memorial Day, we believe we will make substantial progress by the 26th.
2. **Implementation of the EA section**

    a. While two of us complete the opcode section Tarcisius will work on the EA section. Once DJ and Mohammed are done with the opcode section they will help complete the rest of the EA implementation.
3. **Documentation and touch up**
    a. We hope to have time at the end where all of us will comment on each of the subroutines to make it clear what each subroutine was created for. So far we have not gotten to do this as we are pretty focused on the implementation of the opcode
4. **Testing and Debugging**
    a. Deeper testing will be done once more of the opcode and EA sections are completed. Edge cases need to be identified and completed as well during this portion.
5. **Presentation and Demo**
    a. We have yet to start working on the presentation but all of us recognize this needs to be done. We have all agreed to write the presentation on the parts we completed since we will have the best understanding of the sections we completed. This will also give the professor an idea of who completed what sections and demonstrate our understanding.
    b. We plan on using Zoom or Discord to record our presentation and demo.

Problems:

The biggest problem to report is that having 7 data registers forces us to move data back and forth from and to memory or overwriting values, therefore, causing some latency issues. In order to mitigate this, we are considering to also use address registers. Additionally, the stack needs to be more carefully managed as our code includes several nested subroutine calls.

Work Scheduled:

1. Create more detailed models for processes involved in the project **(Complete)**
2. Begin I/O section **(Complete)**
    a. Completed ASCII to Hex
    b. Completed Hex to ASCII

    c. Completed ability to linearly go through memory from inputted memory addresses
3. Implementation of the opcode **(in progress)**
    a. Finish BCC, JSR, RTS, and BRA implementation
    b. Finish up size determining sub-routine.
    c. Add Effective address mode detection
    d. Allocate space in registers for effective address masking
4. Finish I/O section **(Deadline 5/15/2020)**
    a. Needs touch up work and documentation
    b. Rest is for after serious implementation of EA and opcode for priority reasons
5. Implementation of effective addressing **(in progress)**
6. Merge all portions together **(5/27)**
7. Testing and Debugging **(5/27-6/01)**
8. Documentation **(5/27 - 6/01)**
9. Create Presentation **(6/02)**
10. Complete project based on feedback (**6/02-6/07)**

Evaluation:

As the deadline is coming closer each of us feel like we need to contribute more to this project. Now that we have a complete understanding of how vast the project is we really stepped up to the plate to get the work done. To adapt to this, we have decided that two of us will work on the opcode section while the third person completes the EA section. This way we have maximum efficiency.