Program Specification

by Team A: Orpal Nagra, Yuunbum Lim, Gihwan (Kris) Kwon

Program Manual

Our E68k disassembler can decode the supported operations listed below. For a demonstration of how to run the program, please refer to the demo: https://youtu.be/MpUNNG5oPzk

Starting the program

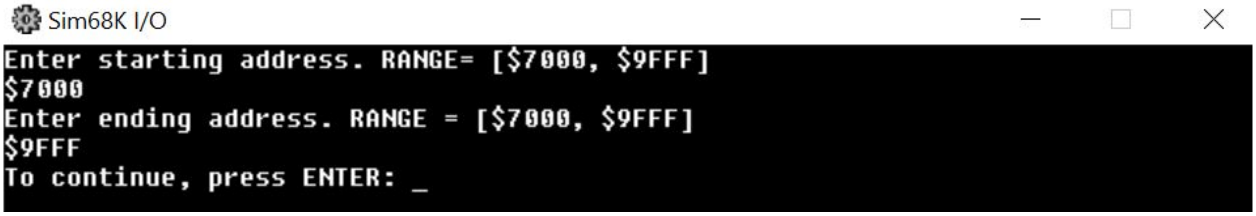
When starting the program, the user selects the X68 file, selects **Assemble Program** from the Project tab in Easy68k. Next, the user can select **Execute** from the menu that appears. This builds the program and prepares it to run. The next step is to add the desired source file- this can be done by selecting the File tab and **Open Data** in the build menu. This acts as the input file which is read by our disassembler (Note: ensure the starting address for the desired file is >= $7000 to ensure all operations are covered). After this, select **Run** and follow on screen prompts.

Input/Output

Upon starting the program, the user is prompted to enter a starting address and ending address within the range of $7000 to $9FFF. When enter is pressed, 20 opcodes are printed in the format:

- 7000 (address) Opcode_Name <ea>, <ea>

Upon each press of enter, 20 opcodes are printed until the program reaches the end of the entered range. If the opcode is not a supported operand and is not recognized, the program will replace the opcode name with “DATA” and the addresses field will be $WXYZ. An example of the starting prompt is shown below:



```
Enter starting address. RANGE= [$7000, $9FFF]
$7000
Enter ending address. RANGE = [$7000, $9FFF]
$9FFF
To continue, press ENTER: _
```

Supported Operations and Addressing Modes

The following is a list of supported operations and addressing modes:

- **Operations supported by disassembler**
  - RTS, NOP
  - JSR, LEA
  - OR, ORI, EOR
  - BCLR
  - CMP, CMPI
  - MOVE, MOVEA, MOVEM
  - NEG
  - MULS, DIVS
  - SUB, SUBQ

  - ADD, ADDA
  - BRA, BCC, BCS, BVC, BGE, BLT
  - ASL, ASR
  - LSL, LSR
  - ROL, ROR

- **Addressing supported by disassembler**
  - Data Register Direct: Dn
  - Address Register Direct: An
  - Address Register Indirect: (An)
  - Address Register Indirect with Post incrementing: (A0)+
  - Address Register Indirect with Pre incrementing: -(SP)
  - Immediate Data: #
  - Absolute Long Address: (xxx).L
  - Absolute Word Address: (xxx).W

Important Notes:

- Input program should reside in the memory from $7000 to $9FFF
- Disassembler reside in the memory starting from $1000
- BRA, BCC, BCS, BVC, BGE, BLT
  - These functions actually calculates absolute address instead of just displaying displacement
- MULS.L and DIVS.L are not currently supported by Easy68k, however we implemented the ability to decode these
- NOP code was not in the requirement, but it is implemented in the disassembler
- ADD code was not implemented in a way that it can handle the source operand with the immediate addressing, because 68K recognize the instruction with immediate addressing source operand as ADDI instruction.
- ADDA was not implemented in a way that it can handle the immediate addressing with a length less than 3 bits, because 68K recognizes the instruction as a SUBQ instead of ADDA when we checked the opcode in the test code.
- 68K reads ADD as ADDI when source operand is an immediate value. (For all size: byte, word, and long)
- 68k reads SUB as SUBI when source operand is an immediate value. (For all size: byte, word, and long)
- 68K reads ADDA with immediate value on source operand as SUBQ.

Team Coding Standards

For successful completion of this project, developing norms and coding standards was essential. Some important aspects of the program are highlighted below:

- Data Register D5 holds the current Opcode bits for each read
- Address Register A5 holds the result message
    - Save this to the stack when decoding each function and recover from stack with newly added chars once a line is decoded
    - When a function is decoded, please utilize post incrementing to add to the result message string
- Address Register A2 points to the current address being read
    - Starts at starting address inputted by user
- Address Register A3 points to the end address entered by the user
    - A2 continually increments until A3 is reached
- The get_bit function can be repeatedly utilized for decoding- it takes inputs D0 (Opcode), D6 (holds value to AND with Opcode in order to achieve a bitmask), and D7 (number to shift right after masking). This is very useful for getting specific pieces of a 16 bit Opcode
- Each character's hex value is referenced at the beginning of the document- this allows ease of adding the characters once they are decoded. Every letter in the English Alphabet can be referenced in the format #Letter_. Numbers 0-9 are also included in this format- (ex. #ZERO_)- in addition to the various special characters outlined below:
    - CR
    - LF
    - NULL
    - TAB
    - DOLLAR
    - COMMA
    - PERIOD
    - PLUS
    - MINUS
    - Open_paren
    - Close_paren
    - SPACE
    - SHARP
    - SLASH
    - DASH
    - COLON