Disassembler Program Specification

Program function

This program is designed to take a binary assembly language code file (.S68 file) written for the Motorola 68000 microprocessor and convert it back into its original assembly source code text. It does not support every instruction that the Motorola 68000 chip offers. The supported instructions are: NOP, MOVE, MOVEA, MOVEM, ADD, ADDA, SUB, SUBQ, MULS, DIVS, LEA, OR, ORI, NEG, EOR, LSR, LSL, ASR, ASL, ROL, ROR, BCLR, CMP, CMPI, Bcc (BCS, BGE, BLT, BVC), BRA, JSR, and RTS. The supported effective addressing modes (when available for that instruction) are: Dn, An, (An), (An)+, -(SP), # (immediate data), (xxx).L (absolute long address), and (xxx).W (absolute word address). Any unsupported function or effective addressing mode found in a program will be displayed as “DATA” along with its hexadecimal value found in the binary file and the program will continue to function.

Use

Upon starting the program, the user is prompted to specify a starting address of the target program to disassemble. The user must enter a value within the range specified in this message and press ‘Enter’. The user must then enter an ending address for the target program (or the end of the section to disassemble). If the two numbers are invalid, the prompts will replay until they receive two valid addresses. If the two numbers are valid, the screen fills with three columns of text. The first column of text indicates the location of the disassembled instruction. The second column is the Assembly Language pneumonic for that instruction (i.e.: MOVE, BCLR). The third column lists any arguments for that function. For example, the decoded instruction MOVE.B D0,D1 found at line 1000 would separate into the three columns as:

00001000 MOVE.B D0,D1

When the user is finished reading the results on the page, they can advance to the next page of decoded instructions by pressing enter again. When the last page of decoded instructions is reached, the user may press the ‘Y’ key to restart the program.

REB's Coding Standards

General

The REB's are Robert Griswold, Eric Mitchell, and Brian Lorton. Throughout the project,Git was used to maintain up-to-date versions of the team's files.

Disassembler

Our program begins with a description of the program function. Given the unclear nature of the assembly language, brief comments were necessary on nearly every single line of code. In addition, each function and complete, distinct block of code is given a larger comment which is separated into a few sections:

- The first section is a brief summary of the code's function and how it is used.
- The next section is a TO-DO section if any work is still left to be done.
- The third section is a NOTES section which indicates any special exceptions or side-effects of calling the function.
- The PARAMS section shows what results from the function and where that information will be placed.
- The RETURN section indicates any value returned by the function, if any. The MODIFIES section indicates any changes to values within registers or memory caused by the function.

Test Code

Since the functions in the test code only need to be reproduced as they appear in the source code and do not collectively accomplish anything meaningful, there are much fewer comments. Comments are given at the beginning of each set of tests for an individual function. For instance, a comment will indicate when we are testing a new function like MOVE. another comment may appear shortly after which indicates MOVE: Dn to Dn.

REB's Test Plan

Test Code Methodology

The test code for all functions is contained within one master file. The goal of our test code was to be exhaustive. When possible, we wrote every possible instance of a function: like

MOVE.B Dn,Dn, which uses every combination of Dn,Dn. Where this was not possible, like with immediates, we chose between 5-12 cases that may be exceptional(-1,0,$FFFF) and used these to test the functions with these numbers.

Early Testing

Any early testing for the disassembler (like NOP and DATA) was done on the disassembler program itself. This type of testing was used for the first and second week of work on the disassembler.

Testing Techniques

The tests for any function were performed during and after the completion of that function. The tests were run by loading the test program into memory, specifying the memory address where the function to test is located, and running the disassembler on this code. Results from the disassembler were compared manually with the original test source code. This process was repeated with the test code starting in several different locations in memory. Occasionally, the test code for any function is checked again to make sure that any new additions to the code did not affect the proper functioning of old blocks of code.

Task Breakdown Report

At the beginning of the project, we assigned roles to each of our three team members. These roles were as follows:

- Robert Griswold - Decode functions and disassembler framework
- Brian Lorton - Decode functions and disassembler framework
- Eric Mitchell - Test code and documentation.

Team members filled in where needed or where was necessary to have complete understanding of the disassembler. For example, Brian Lorton created the test code for our disassembler presentation--a more concise version of the exhaustive test code which better suited a short presentation. Eric Mitchell decoded a few functions for the disassembler to make sure he understood its inner workings. The breakdown of work is as follows:

Task	Implementer	Time to complete
Disassembler Design	Robert Griswold	1 day
Disassembler Main	Robert Griswold	1/2 day
Input Subroutine	Robert Griswold	2 days
Effective Address Subroutine	Robert Griswold	1 day
Output Subroutine	Robert Griswold	1 day
Hex Conversion Subroutine	Robert Griswold	1/2 day
Test Code	Eric Mitchell	1.5 weeks
Decoder Flowchart	Brian Lorton	1.5 weeks?
NOP	Robert Griswold	15 mins
MOVEM	Eric Mitchell	30 mins
MOVE/MOVEA	Eric Mitchell/Eric Mitchell	2 hours/15 minutes
ADD/ADDA	Brian Lorton	20 hours :(

SUB/SUBQ	Brian Lorton/Eric Mitchell	2 hours / 3 days
MULS/DIVS	Brian Lorton	30 mins / 30 mins
LEA	Eric Mitchell	30 minutes
OR/ORI	Brian Lorton	3 hours / 3 hours
NEG	Robert Griswold	1/2 day
EOR	Brian Lorton	1 hour
LSR/LSL/ASR/ASL/ROL/ROR	Robert Griswold	1 day
BCLR	Brian Lorton	3 hours and counting
CMP/CMPI	Brian Lorton	1 hour / 2 hours
BCC	Brian Lorton	6 hours
BRA/JSR	Brian Lorton/Eric Mitchell	1 hour/10 seconds
RTS	Eric Mitchell	1 hour
Project Description	Eric Mitchell	1 day
Program Specification	Eric Mitchell	1 day
Test Plan	Eric Mitchell	1 day
Exceptions Report	Eric Mitchell	1 day
Task Breakdown Report	Eric Mitchell	1 day