Team 10: Norman Hansen, Konstantin Stekhov

User Guide

1) Load the test data file in the memory of the disassember in Easy68k
2) Execute the disassember file
3) Input valid starting and ending addresses and you will see that the disassembled instructions are printed
4) When all of the instructions are printed and you can see the end of the program message close the console window
5) You may execute a program again with a different test data file

The Program Specifications

1) Program starts
2) Display the program description message
3) Program prompts for the starting address
4) Program reads the input and converts it from ASCII to HEX
5) Check if the starting address has a valid size (between 0 and 8 characters) and reprompt a user if it is not the case
6) Check if the starting address is even and if it is odd then reprompt a user for a new address
7) Check if the starting address is between 7000 and FFFFFE and if not then reprompt for a new address
8) If the starting address is valid then put it into A4
9) Program prompts for the ending address
10) Program reads the input and converts it from ASCII to HEX
11) Check if the ending address has a valid size (between 0 and 8 characters) and reprompt a user for new starting and ending addresses if it is not the case
12) Check if the ending address is even and if it is odd then reprompt a user for a new starting and ending addresses
13) Check if the ending address is between 7000 and FFFFFE and if not then reprompt for new starting and ending addresses
14) Check if the ending address is larger than the starting address if not then prompt a user for new starting and ending addresses
15) If the ending address is valid then save it in A5
16) Then the program starts executing the data loop by getting a word from the address stored in A4
17) Op code command stored in D3 gets the first 4 bits and they go to D1 for comparison
18) The program checks if the instruction is valid and if it is not valid then the program adds an error to the buffer in A1 and prints it then increments an address in A4 and then decodes the first 4 bits of the instruction and goes to next step if it is valid

19) Then when the part of the instruction is matched the program decodes the rest of the instruction
20) Next the program checks if the instruction has an effective address, immediate data, or displacement
21) If it is the case then the program increments the address in A4 by a word and decodes the ea mode which is in D6 or address mode which is in D7
22) After the instruction is decoded its characters are translated from HEX to ASCII and then the ASCII characters are added to the buffer in A1 and printed using PRINT subroutine
23) Check if the address in A4 is less than address in A5 if yes then process the next instruction if not then the program ends
24) The program displays the end message

Team Code Standards

- User uppercase letters and words for all of the instruction and variables that are in the code
- Comment the code so everyone understands what is happening

Instructions

MOVE, MOVEA, MOVEM
SUB, SUBQ
ADD, ADDA
LEA
MULS, DIVS
OR, ORI
NEG
ASL, ASR
LSR, LSR
ROR, ROL
BCLR
CMP, CMPI
Bcc (BCS, BGE, BLT, BVC)
BRA, JSR, RTS

Effective Address Modes

Address Register Direct
Address Register Indirect
Address Register Indirect with Pre Incrementing
Address Register Indirect with Post Decrementing
Data Register Direct
Immediate Data
Absolute Long Address

Absolute Word Address