DESIGN PROJECT - PART 1

1.0 Purpose

The purpose of this experiment is to accomplish the following:

- Learn how to download programs from a host computer into the SANPER-1 ELU.
- Understand the architecture of an interactive debugger
- Implement a proper command interpreter

2.0 Component Requirements

None.

3.0 Background

A. Downloading Capability

Through a combination of hardware and software, the SANPER-1 ELU is capable of receiving MC68000 programs from an external device (i.e. computer) then storing these programs into the SANPER-1 ELU's memory. This downloading capability is achieved in hardware by connecting the serial port of the computer to one of the serial ports of the SANPER-1 ELU. The download functionality is achieved in software through the TUTOR firmware. Invoking TUTOR's Transparent Mode Command ("TM") sets up the SANPER-1 hardware to wait for data to arrive through one of its serial ports. The external computer then transmits a file out of its serial port. The file is sent in Motorola S-Record format. The TUTOR firmware reads in the data from its serial ports and stores it into memory.

The procedure to download a program from a personal computer to the SANPER-1 ELU is described in the SANPER-1 Educational Lab Unit User's Manual.

B. Debugger

In this lab the Debugger refers to a program that helps developers to exam certain information or performs a certain task on a computer powered by MC68000 CPU. As TUTOR, the Debugger should have the capability to take user input and response accordingly. For instance, when using the TUTOR, typing in "DF" the computer will display the content of all registers. To implement a Debugger, following aspects need to be considered:

- ***Proper architecture:*** The debugger needs to be modular. Each debugger command should be implemented as a subroutine. Thus it will not interfere with other commands. Some utility functions should be implemented and shared across

debugger commands to reduce redundant code.

- ***Robustness:*** In the event of an exception, the debugger should be able to provide the necessary information (i.e., BA IR SSW) for debugging and recover the computer.

- ***Command Interpreter***: Parse the input from the user, then branch to the proper subroutine.

- ***Debugger command:*** First, you need to clearly define the function and syntax of each debugger command. Then implement each command to parse the input and understand the operant and mode of operation when applicable. Finally, implement the rest of the command so it can perform the operation according to user input.

- ***User friendliness****:* When user input incorrect command, the system should remain intact and provide the user with an example of correct syntax and specification of each command. Note, this is different from exception handling.

4.0 Statement of Problem

In the first part of this experiment, the student will download a sample program to the SAMPER-1 ELU and then execute it. This exercise will provide the student with practice in downloading programs to the SANPER-1 ELU.

In the second part of this experiment, the student will use the EASy68K to implement Debugger. Parts of the codes have been provided as an example for implementing the Debugger.

5.0 Preliminary Assignment

A. Sample Downloading Program

1. Create a file on the host computer and enter the program listed in Table 4.1. Some assembler-specific directives may need to be added to the program for it to be assembled by the host computer's MC68000 cross-assembler. A list of directives is available in the User's Manual for the specific assembler.

2. Assemble the program using the appropriate host computer commands. The assembled program should be free of syntax and assembler errors. The MC68000

cross-assembler will generate two output files: a LIST files and a TAPE file. The TAPE file consists of Motorola S-Records and is the file which will be downloaded to the SANPER-1 ELU. The LIST file is a listing of the assembled source code. Bring two copies of each file to the lab.

3. Read Appendix A of the MC68000 Educational Computer Board User's Manual, entitled "S-Record Output Format" which discusses the Motorola S-Record data format. Closely examine the S-Records in your TAPE file. You should be able to identify each of the six fields within each record. Copy these records onto another sheet of paper, label each field and explain its function.

B. Debugger

1. Assemble the example code (Design Project Part1.x68) in EASy68K and run some examples.

2. When you type SORTW $10000 $10010 A, what subroutines been called? Draw a flow chart explaining the order in which each subroutine been called.

3. When performing SORTW $10000 $10010 A, what information (i.e., address number etc.) been passed between subroutines and how (i.e., RAM STACK or Register)? Show the process step by step, to be more specific, from the user type in the SORTW command till the computer print the result.

4. Exam the help command, some of the help information been implemented, please go ahead and finish the remaining ones.

5. Implement all function specified. You should use existing subroutines in the example code (distributed separately). No TRAP function shall be used. Your function should be able to discover syntax error and display a syntax error message, as shown in the examples.

    a. SORTW: The SORTW command sorts a block of memory. The starting address <address1> and the ending address <address2> of the memory block are specified in the command. The size of the data to be sorted is a word. The order (A or D) specifies whether the list is sorted in Ascending or Descending order. This command should not interfere with other parts of the program.
    Examples:
    SORTW $3000 $3200 A -> Sort successful message
    SORTW $3200 $3000 A -> Syntax error message
    SORTW $3000 $3200 F -> Syntax error message

    b. AND: input two hex number then conduct bitwise AND operation and print the result in hex
    Examples:
    AND $55 $0033 -> $00000011
    AND $55 $0033 -> Syntax error message
    AND $5H $0033 -> Syntax error message

c. OR: input two hex number then conduct bitwise OR operation and print the result in hex
Examples:
OR $0000 $F1 -> $000000F1
OR $0000 $F1 -> Syntax error message
OR $0000 $J1 -> Syntax error message

6.0 Procedure

1. Download the sample program from Table 4.1 to the SANPER-1 ELU using the downloading procedure outlined in the last page of Lab 2 Instruction Manual.
2. Execute the sample program from Table 4.1 on SANPER-1 ELU.
3. Record the results of running this program.
4. Run the Debugger implemented by you in Easy68K.
5. Test the subroutines with varies case, fix bug(s).
7. Demonstrate to your Lab Instructor that your program works correctly.

7.0 Discussion

Submit the following items to your Lab Instructor as a Design Project Part 1 Report:

1. A listing of your Debugger program with both global and local comments.
2. A description of the S-Record fields for the sample program (from Table 4.1).
3. Draw a block diagram and illustrate in detail how ASCII encoded hexadecimal numbers are converted to hex encoded numbers in Design Project Part1.x68? (Which registers are used to pass input value(s), boundary condition, exception, etc.)
4. Draw a block diagram and illustrate in detail how SORTW subroutine was implemented? (Which registers are used to pass input value(s), boundary condition, exception etc.)

TABLE 4.1 Sample Downloading Program

```
            ORG $900
MSGSTR      DC.B 'IT WORKS!'
MSGEND      DS.B 1

            ORG $1000
START       LEA $2FFE,A7
LOOP        LEA MSGSTR,A5
            LEA MSGEND,A6
            MOVE.B #243,D7
            TRAP #14
            MOVE.B #241,D7
            TRAP #14
            MOVE.B #227,D7
            TRAP #14
            BRA LOOP
            SIMHALT
            END START
```