EXPERIMENT #1

INTRODUCTION TO THE SANPER-1 EDUCATIONAL LAB UNIT

1.0 Purpose

The purpose of this experiment is to introduce the user to the following items:

- the SANPER-1 Educational Lab Unit
- the TUTOR Command Set
- the TUTOR TRAP 14 Handler
- the MC68000 Instruction Set

The purpose of this exercise is to introduce the user to some of the basic commands offered by TUTOR and to familiarize the user with the manner in which TUTOR operates.

2.0 Component Requirements

None.

3.0 Background

A. Description of the SANPER-1 Educational Laboratory Unit

The SANPER-1 Educational Unit was designed and developed by Dr. J. Saniie and Mr. Stephen Perich. The SANPER-1 system is equipped with all the hardware and software required for implementing a complete MC68000 based development system.

Hardware Features

The SANPER-1 ELU is equipped with the following devices:

- an MC68000 Microprocessor
- 208K bytes of static RAM
- 32K bytes of EPROM
- three serial ports
- two parallel ports
- an A/D converter
- a D/A converter
- a Speech Synthesizer
- two high power output drivers.

Special features of the SANPER-1 ELU include : the Status Display, the Single Step Hardware Mode, the Parallel Printer Interface, the Cassette Recorder Interface, and three different methods for interfacing external hardware to the unit.

Software Features

The SANPER-1 ELU contains the TUTOR Monitor program which is discussed in the next section.

B. Description of the TUTOR Resident Monitor

TUTOR is the resident 16K-byte firmware package that provides self-contained programming, operating, and debugging environment. TUTOR interacts with the user through predefined commands that are entered via a terminal. There are five general categories of TUTOR commands:

a. Commands which allow the user to display and change memory.

b. Commands which allow the user to display or modify the internal registers of the 68000.

c. Commands which allow the user to execute a program.

d. Commands which allow the user to debug programs.

e. Commands which allow the user to interface to other types of equipment. (ie. host computers, printers, audio cassette recorders, etc.)

Table 3-1 of the MC68000 Educational Computer Board User's Manual lists the available TUTOR commands. Refer to this manual for a detailed description of each TUTOR command.

The most frequently used commands are the following:

- Display Formatted Registers (DF). This is used to display the internal registers of the 68000.
- Memory Modify (MM). This command is used to enter your program into the lab unit.
- Memory Display (MD). This command is used to examine the contents of memory.
- Go (G). This command is used to execute your program.
- Trace (T). This command is used during debugging to step through your program one instruction at a time.

C. TUTOR'S 1-line Assembler/Disassembler

TUTOR is equipped with a 1-line assembler/disassembler, which allows the user to enter his or her source code directly from a terminal.

The disassembler can be used with either the Memory Modify (MM) or Memory Display (MD) commands. The difference between the two commands is that the MD command can only be used to disassemble one instruction, while the MM command can be used to disassemble an unlimited number of instructions.

To invoke the disassembler, add the suffix ";DI" immediately after the address when using the Memory Modify or Memory Display commands.

Example: `TUTOR 1.X > **MM $900;DI <CR>**`

TUTOR will respond as follows:

```
000900   FFFF                DC.W  $FFFF ?
```

The above line consists of three fields: the address (000900), the assembled version of the instruction (FFFF), and the unassembled version of the instruction (DC.W $FFFF). The instruction displayed on your terminal may or may not be "DC.W $FFFF", but this is irrelevant since we are concerned only with the format for the present time.

To enter a new instruction, simply move the cursor over one space, and type in your instruction. You must space over ONCE or TUTOR will not accept your input.

An example of the correct format is:

```
000900   XXXX                DC.W  $FFFF ? MOVE.W D0,D1 <CR>
```

An example of the incorrect format is:

```
000900   XXXX                DC.W  $FFFF ?MOVE.W D0,D1 <CR>
```

If the user doesn't space over at least once, or if a typographical or syntactical error is made, TUTOR responds with the following:

```
                X?
```

This prompt indicates that TUTOR did not accept the instruction that was entered. If this prompt appears, move the cursor over one space and reenter the instruction.

```
X? MOVE.W D0,D1 <CR>
```

Once TUTOR has accepted the instruction, it immediately assembles it and displays the new instruction in place of the old one.

```
000900   3200                MOVE.W D0,D1 ?
```

TUTOR then displays the next instruction, and the user repeats the editing procedure.

```
000902   XXXX                ADDQ.W #1,D2 ?
```

Once the user has entered his entire program into the system, he terminates editing by typing a period (.) immediately after the question mark of the next instruction following the end of his program.

```
009004   XXXX                SUBQ.B #1,D7 ?. <CR>
```

TUTOR then displays its prompt, and waits for the user to enter a new command.

```
TUTOR 1.X >
```

D. TRAP 14 Handler

Another very useful feature contained within TUTOR is the TRAP 14 Handler. The TRAP 14 Handler is a function that allows user programs to invoke system utility programs. In other words, the TRAP 14 Handler allows the user access to TUTOR's library functions and routines. For instance, if the user wishes to write a character to the terminal, he/she can simply call the utility function that handles all the initialization and programming of the ACIA.

E. TRAP 14 Calling Sequence

The user program invokes TUTOR's TRAP 14 Handler with the following calling (command) sequence:

```
MOVE.B #<Function Number>,D7
TRAP #14
```

The field <Function Number> is a decimal number from 0 to 255 which identifies the specific system routine that is to be executed. Table 5-1 of the MC68000 Educational Computer Board User's Manual lists the available TRAP 14 functions. For a detailed description of each TRAP routine, refer to Chapter 5 of the 68000 Educational Computer Board User's Manual.

Any of the functions within the TRAP 14 Handler can be called at any time, but the user must ensure that any registers or memory locations are initialized for that specific function before calling the TRAP 14 Handler.

The following example illustrates the use of TUTOR's TRAP 14 Handler. We will use the TRAP 14 Handler's Function #248 to output a character to the terminal. Note

that the character to be outputted must be placed into data register D0 before the TRAP 14 Handler is called.

```
ADDQ.W #$10,D3              * any 68000 instruction
.                           *       ”
.                           *       ”
MOVE.B #$35,D0              * Output the number ’5’ to
MOVE.B #248,D7              * Port #1 (terminal) using
TRAP #14                    * TRAP function #248.
.                           * any 68000 instruction
.                           *       ”
.                           *       ”
```

The TRAP 14 Handler will be used extensively in later laboratories.

4.0 Statement of Problem

In this experiment the user will be introduced to the SANPER-1 Educational Lab Unit’s TUTOR monitor program. The user will gain familiarity with the TUTOR commands, and will enter several programs into the lab unit using an input device such as a terminal or host computer. Several sample programs are presented that may require modifications. The user will be entering, executing and debugging these programs, and answering questions that are raised throughout the programs.

5.0 Preliminary Assignment

1. Review the TUTOR resident monitor commands, which are listed in Tables 3-1 and 3-2 of the MC68000 Educational Computer Board User’s Manual. Although the user is not required to memorize these commands, he/she should be familiar with their function.

2. Review the 68000’s instructions and addressing modes. The user should read both the ”Data Organization and Address Capabilities” section and the ”Instruction Set Summary” section in the MC68000 program’s Reference Manual.

3. Read the SANPER-1 Educational Lab Unit User’s Manual to gain familiarity with the lab unit. A procedure is included for setting up the SANPER-1 ELU for Hardware single Step Mode.

6.0 Procedure

PART A - Instruction to the TUTOR Command Set and the SANPER-1 ELU

This part of the procedure will introduce the student to some of the basic TUTOR Resident Monitor commands.

1. Invoking the HELP Command (HE)

Type in the following command:

```
TUTOR 1.X > HE <CR>
```

TUTOR displays the following information:

```
.PC   .SR   .US   .SS
.D0   .D1   .D2   .D3   .D4   .D5   .D6   .D7
.A0   .A1   .A2   .A3   .A4   .A5   .A6   .A7
.R0   .R1   .R2   .R3   .R4   .R5   .R6

BF    BM    BR    NOBR   BS    BT    DC    DF
DU    G     GD    GO     GT    HE    LO    M
TM    TR    TT    VE
```

The top half of the Help Screen lists all the registers available for user programming. All of the registers except Offset Registers R0 through R6 are internal registers of the 68000. The user is allowed to read and write to each of these registers. TUTOR provides the Offset Registers for ease of program relocation.

The bottom half of the Help Screen lists the TUTOR instructions that allow the user to enter, run, and debug programs in 68000 Assembler. Also included are system-level instructions that allow the user to upload and download files to a host computer, print out the contents of memory, etc.

All of the instructions and their descriptions are listed in Table 3-1 on page 3-9 of the MC68000 Educational Computer Board User's Manual.

2. Invoking the Display Formatted Registers Command (DF)

Type in the following command:

```
TUTOR 1.X > DF <CR>
```

TUTOR displays the following information:

```
PC=00001000  SR=2700=.S7.....  US=00002000  SS=00000F00
D0=00000010  D1=00007FFF  D2=FFFF0000  D3=000003FF
D4=BBBBFFFF  D5=0000A0A0  D6=00007777  D7=0000FFFF
A0=00010040  A1=00008146  A2=00001000  A3=00010000
A4=00000555  A5=0000A000  A6=00008765  A7=00001200
--------------------------001000     2248          MOVE.L A0,A1
```

All of the registers shown above are internal registers of the 68000, and are described as follows:

Quantity	Description
1	Program Counter (PC)
1	Status Register (SR)
1	User Stack Pointer (US)
1	Supervisor Stack Pointer (SS)
8	Data Register (D0 to D7)
8	Address Registers (A0 to A7)

The contents of each registers is displayed in hexadecimal notation (base 16). All registers are 32 bits wide except for the Status Register, which is 16 bits wide.
The bottom-line if the display shows the next instruction to be executed, in both its disassembled and assembled forms.

3. Displaying and Modifying the Status Register

Notice that to the right of the hexadecimal listing of the Status Register contents is an 8-character field. The meaning of this field is as follows:

```
SR=XXXX=(T)(S)(I2 I1 I0)(X)(N)(Z)(V)(C)
FIELD:   1  2      3     4  5  6  7  8
```

Where each field is defined as follows:

Field	Abbreviation	Description
1	T	Trace Mode Bit
2	S	Supervisor State Bit
3	I2, I1, I0	Interrupt Mask
4	X	Extend Bit
5	N	Negative Bit
6	Z	Zero Bit
7	V	Overflow Bit
8	C	Carry Bit

For all cases except the Interrupt Mask, whenever any of the bits are set, the alphabetic character appears in the listing. For example, if the S bit is cleared, the display will read as follows:

```
SR=2700=.S7.....
```

When the bit is cleared (not set), a period (.) appears in the display. For example, if the S bit is cleared, the display will read as follows:

```
SR=0700=..7.....
```

The Interrupt Mask is a 3-bit value, which represents the current interrupt level of the 68000. Rather than displaying all three bits in the expanded version of the Status Register, TUTOR has simplified the display of this 3-bit mask into a single integer whose value ranges from 0 to 7. Thus if a level 7 interrupt has just been received by the 68000 TUTOR displays the interrupt mask as 7.

```
SR=0700=..7.....
```

This extra 8-character field in the Status Register provides a very quick way of determining which Status Register bits are set.

Note that the reason the user may freely change the bits in the Status Register is that TUTOR is executing in the Supervisor Mode. Typically while executing a program the user is operating in User Mode, and the user is prohibited from modifying the Status Register. Any attempts to do so will result in a Privilege Violation Exception.

Example No. 1: Setting the Status Register to 0000.

In this example we will use the individual register display / change command to change the contents of the Status Register to 0000.

Modify the Status Register by typing in the following command:

```
TUTOR 1.X > .SR 0000
```

The Status Register has been modified to set all bits equal to zero.

Examine the Status Register by typing in the following:

```
TUTOR 1.X > DF <CR>
```

TUTOR will display all of the 68000's internal registers. The Status Register will appear as follows:

```
SR=0000=..0.....
```

Notice that since all bits are equal to zero, this is represented as periods for all alphabetic bits, and as a zero for the Interrupt Mask.

Example No. 2: Setting the Status Register to $FFFF.

In this example we will use the individual register display / change command to change the contents of the Status Register to $FFFF.

Modify the Status Register by typing in the following command:

```
TUTOR 1.X > .SR FFFF
```

The Status Register has been modified to set all bits equal to one.

Examine the Status Register by typing in the following:

```
TUTOR 1.X > DF <CR>
```

TUTOR will display all of the 68000's internal registers. The Status Register will appear as follows:

```
SR=FFFF=TS7XNZVC
```

Notice that since all bits are equal to one, this is represented as the actual character for all alphabetic bits, and as a seven for the Interrupt Mask.

4. Individual Register Display / Change Commands

Each of the other internal 68000 register (Address, Data, Program Counter, and Stack Pointers) can be displayed or modified in exactly the same manner as previously shown for the Status Register.

Example No. 1: Changing the contents of an Address Register.

Modify Address Register A1 by typing in the following:

```
TUTOR 1.X > .A1 1234 <CR>
```

This command changes the content of the Address Register A1 to $00001234.

Examine this register by entering the following command:

```
TUTOR 1.X > .A1 <CR>
```

TUTOR will display the following:

```
A1=00001234
```

Example No. 2: Display all address registers.

In order to examine the eight address registers simultaneously, enter the following command:

```
TUTOR 1.X > .A <CR>
```

The eight address registers will be displayed as follows:

```
A0=00000000 A1=00000000 A2=00000000 A3=00000000
A4=00000000 A5=00000000 A6=00000000 A7=00000000
```

Example No. 3: Display all data registers.

In order to simultaneously examine the eight data registers, enter the following command:

```
TUTOR 1.X > .D <CR>
```

The eight data registers will be displayed as follows:

```
D0=00000000 D1=00000000 D2=00000000 D3=00000000
D4=00000000 D5=00000000 D6=00000000 D7=00000000
```

PART B – Entering, Executing, and Debugging several Sample Programs

In this part of the procedure the student will enter several MC68000 Assembly Language programs into the SANPER-1 ELU using TUTOR's 1-line assembler. Once the programs are correctly entered, they will be executed and debugged.

1. Type the unassembled source code (the actual 68000 instructions) of Table 1.2 into the lab unit using TUTOR's "Memory Modify" (MM) Command. Invoke the disassembler so that the assembly language mnemonics can be entered directly. Do not enter any 68000 cross-assembler directives. Start your program at location $1000. The TUTOR commands should resemble the following:

   ```
   TUTOR 1.X > MM $1000;DI
   ```

   TUTOR will respond as follows:

   ```
   001000   XXXX          XXXXXXXXX ?
   ```

   Move the cursor over one space, then enter the first instruction and follow it with a carriage return, <CR>, as follows:

   ```
   001000   XXXX          XXXXXXXXX ? LEA $2000,A7
   ```

Continue entering the entire program one instruction at a time until all are entered. Once all instructions have been entered, type a period (.), and then a carriage return <CR> to return to the TUTOR prompt.

```
001XXX   XXXX          XXXXXXXXX ?. <CR>
```

2. Enter the following command:

```
TUTOR 1.X > MS $900 'IT WORKS !!' <CR>
```

3. Run the program using TUTOR's "Go" (G) command. The format is as follows:

```
TUTOR 1.X > G $1000 <CR>
```

What happens ever time a message (or any text) and a <CR> is typed in at the keyboard? If your program gives no response on the terminal, then it ins't working properly, so go back and examine the source code using the "Memory Modify" command.

4. To exit your program depress the ABORT switch on the Front Panel of the ELU. What happens when the switch is depressed? Record any information displayed on the terminal.

5. The program listing in TABLE 1.1 is unacceptable because it lacks comments. Repeatedly trace through the program until you become familiar enough with it to rewrite the program adding meaningful global and local comments.

Tracing through a program is accomplished by using TUTOR's 'TRACE' command. Begin by setting the Program Counter to the address where you wish to start tracing, and then enter the Trace (T) command. The format of these instructions is as follows:

```
TUTOR 1.X > .PC $1000 <CR>

TUTOR 1.X > T <CR>
```

While in Trace mode, record the content of each register after each instruction has executed. Record any other data which appears on the terminal. Record and comment on the events that occur while tracing through the TRAP instruction. Stop tracing after the first TRAP instruction. Stop tracing after the first TRAP instruction has been encountered.

Notice that while you are in Trace Mode, the TUTOR prompt changes to:

```
TUTOR 1.X :>
```

To continue tracing through the program, simply enter a carriage return (<CR>) as follows:

```
TUTOR 1.X :> <CR>
```

To exit Trace Mode simply type in a period followed by a carriage return, as follows:

```
TUTOR 1.X :> . <CR>
```

6. Demonstrate to you Lab Instructor that your program works properly.

7. Using the TUTOR commands, type in the sample program listed in Table 1.2. Place the SANPER-1 ELU in hardware single-step mode, and single-step through the program. For each bus cycle, record the status of the Address, Data, and Control lines by examining both the 7-segment alphanumeric displays and the bar-graph LEDs on the front panel of the SANPER-1 ELU.

8. Using the TUTOR commands, type in the sample program listed in Table 1.3. Place the SANPER-1 ELU in hardware single-step mode, and single-step through the program. For each bus cycle, record the status of the Address, Data, and Control lines by examining both the 7-segment alphanumeric displays and the bar-graph LEDs on the front panel of the SANPER-1 ELU.

9. Using the TUTOR commands, type in the sample program listed in Table 1.4. Place the SANPER-1 ELU in hardware single-step mode, and single-step through the program. For each bus cycle, record the status of the Address, Data, and Control lines by examining both the 7-segment alphanumeric displays and the bar-graph LEDs on the front panel of the SANPER-1 ELU.

10. Place the SANPER-1 ELU in its hardware single-step mode. Depress the RESET pushbutton. The Address lines on the ELU Front Panel Display should read $000000.

11. Depress the SINGLE STEP PULSE pushbutton once. Examine the front panel display and record the values of the Address and Data lines, and the state of the Control lines. Repeat this procedure eight times, and then return the ELU to Free Run Mode.

7.0 Discussion

Submit the following to your Lab Instructor as a Final Report:

1. Completed listings of all the program segments. Your listings must include meaningful global and local comments.

2. What is the address range of the available memory within the SANPER-1 ELU that may be used for your program? Describe your answer.

3. For Procedures #7, #8, and #9 list the value of the address, data, and control lines, and discuss any unusual events that may have occurred.

4. How many serial ports are available on a SANPER-1 ELU? What are the specific memory addresses associated with these ports?

5. List all the possible ways to interface you hardware design to the SANPER-1 ELU.

6. When the MC68000 control signals "*AS", "*UDS", or "*LDS" are asserted, what is the status of the respective LED (on or off), and why?

TABLE 1.1 Sample Program No. 1

```
TUTOR 1.X > MM $1000;DI <CR>
```

Address	Instruction	Comments
001000	**LEA.L $2000,A7 <CR>**	To be determined
001004	**MOVE.L #$900,A5 <CR>**	
00100A	**MOVE.L $90B,A6 <CR>**	
001010	**MOVE.B #243,D7 <CR>**	
001014	**TRAP #14 <CR>**	
001016	**MOVE.B #241,D7 <CR>**	
00101A	**TRAP #14 <CR>**	
00101C	**MOVE.B #227,D7 <CR>**	
001020	**TRAP #14 <CR>**	
001022	**BRA $1004 <CR>**	
001024	**. <CR>**	

Table 1.2 Sample Program No. 2

```
TUTOR 1.X > MM $900;DI <CR>
```

Address	Instruction	Comments
000900	**MOVE.B D0,D1 <CR>**	To be determined
000902	**MOVE.B #$AA,$1000 <CR>**	
000908	**BRA $900 <CR>**	
00090A	**. <CR>**	

Table 1.3 Sample Program No. 3

```
TUTOR 1.X > MM $900;DI <CR>
```

Address	Instruction	Comments
000900	**MOVE.B D0,D1 <CR>**	To be determined
000902	**MOVE.B #$AA,$1001 <CR>**	
000908	**BRA $900 <CR>**	
00090A	**. <CR>**	

Table 1.4 Sample Program No. 4

```
TUTOR 1.X > MM $900;DI <CR>
```

Address	Instruction	Comments
000900	**MOVE.B D0,D1 <CR>**	To be determined
000902	**MOVE.B $1000,$1001 <CR>**	
000908	**BRA $900 <CR>**	
00090A	**. <CR>**	

PROCEDURE FOR SINGLE STEPPING THROUGH A PROGRAM USING

THE SANPER-1 UNIT'S HARDWARE SINGLE-STEP MODE

1. Change the exception vector for Autovector Interrupt Level 7 in the exception vector Table to the starting address of the program you wish to single step through.

```
TUTOR 1.X > MM $7C;L <CR>

0000007C ? (Change to the starting address of the program)
00000080 ? . <CR>

TUTOR 1.X >
```

2. Move the "Single Step" / "Free Run" Switch on the front panel of the SANPER-1 ELU up to the "Single Step" position.

3. Verify that the "S.S. EXT" / "S.S. SYS" Switch is in the "S.S. SYS" position.

4. Press the "ABORT" Switch in and HOLD it.

5. Press the "Single Step Pulse" Switch several times until the 68000 address displayed on the front panel 7-segment LEDs is $00007C (about 6 times). The data displayed should be the starting address of your program.

6. Release the "ABORT" Switch

7. Press the "Single Step Pulse" Switch once to advance to the next bus cycle. Continue pressing this switch to advance through your program.

8. To return to the Free Run Mode move the "Single Step" / "Free Run" Switch down to the "Free Run" position.

Notes

1. Step #1 must be repeated after a power-up condition or after pressing the "RESET" button since the exception vector will be changed to its default value.

2. This method changed the processing mode from User to Supervisor.

PROCEDURES FOR INTERFACING A HOST PC TO THE SANPER-1 ELU

PROCEDURE TO ASSEMPLE A FILE ON THE HOST PC

```
C:\WORK> asm -l filename <CR>
```

This command produces two files:

`filename.h68`	Tape file (S-Records)
`filename.lis`	Listing file

PROCEDURE TO CONNECT TO THE SANPER-1 ELU

```
C:\> ee441 <CR>

C:\WORK> KERMIT <CR>

Kermit-MS> C <CR>
```

Hit <CR> a few times until the TUTOR prompt appears. If the prompt does not appear, verify port and baud rate settings.

PROCEDURE TO DOWNLOAD AN S-RECORD FILE TO THE SANPER-1 ELU

```
TUTOR 1.X > LO1

<CTRL> ]

C

Kermit-MS> TRANSMIT filename

Kermit-MS> C <CR>
```

Hit <CR> a few times until the TUTOR prompt appears.

SUPPLEMENTAL NOTES FOR PC SIMULATOR & ASSEMPLER

Simulator

- To input or output a character to the screen use subroutines located at $10000 and $10002, respectively.

  Inputting a character from the screen, e.g.

  JSR $10000

This will input the ASCII character into the lowest byte of register D0.

Outputting a character to the screen, e.g.

JSR $10002

This will print the ASCII character of the lowest byte of register D0.

- Valid memory for program is $0000-$7FFFF, but remember that Exception Vectors reside in 0-C0.

Assembler

- Address Range of programs must be between $0000-$3FFE for TUTOR's use. Other software is available in lab that will compile programs residing in other address ranges.