EXPERIMENT #3

EXCEPTION PROCESSING and SYSTEM CONTROL

1.0 Purpose

The purpose of this experiment is to acquaint the student with the following topics:

a. 68000 Exception Processing
b. TUTOR Exception Handling
c. 68000 System Controls

2.0 Component Requirements

None.

3.0 Background

Review the following material:

a. the Exceptions Processing section of the M68000 Programmer's Reference Manual
b. the SANPER-1 User's Manual

4.0 Statement of Problem

Three sample programs are presented which are to be entered and executed. Numerous questions are raised with regard to exception processing.

5.0 Preliminary Assignment

1. Read all three programs and familiarize yourself with their objectives.

2. Procedure #12 of Sample Program 3.2 requires the student to write an MC68000 Assembly Language Program. This program should be written prior to attending the lab.

6.0 Procedure

Execute each sample program and record data when requested.

7.0 Discussion

Answer any questions that were asked in each sample program, and submit your answers as a final report to your Lab Instructor.

SAMPLE PROGRAM 3.1 - Exception Processing

A. Address Error Exception (Exception Vector No. 3)

1. Enter the following program:

```
TUTOR 1.3 > MM $2000;DI <CR>
```

Address	Instruction	Comments
002000	MOVE.W D0,A1 <CR>	To be determined
002002	MOVE.W D1,(A1)+ <CR>	
002004	BRA $2000 <CR>	
002006	. <CR>	

2. Initialize Data Register D0 to $FF

```
TUTOR 1.3 > .D0 $FF <CR>
```

3. Run the program using the GO (G) Command.

```
TUTOR 1.3 > G $2000 <CR>
```

4. TUTOR responds with the following:

```
PHYSICAL ADDDRESS=00002000

32C5 000000FF 32C1
ADDR TRAP ERROR
```
(All 68000 internal registers)
(Next instruction to be executed)

5. The three fields above the expression "ADDR TRAP ERROR" indicate where the exception occurred. The left-most field indicates the status of the MC68000 when the exception occurred. Only the five least significant bits of this field are ever modified. For further information, refer to Section 4.3.5 on page 4-30 of the MC68000 Educational Computer Board User's Manual (mecb_chapter_4b.pdf). The middle field indicates the address at which the fault occurred, and the right-most field is the assembled instruction at which the fault occurred.

6. Describe how and why the above Address Trap exception occurred.

B. Bus Error Exception (Exception Vector No. 2)

1. Enter the following program:

```
TUTOR 1.X > MM $2000;DI <CR>
```

Address	Instruction	Comments
002000	**MOVE.B $FFFFFF,D0 <CR>**	To be determined
002006	**BRA $2000 <CR>**	
002008	**. <CR>**	

2. Initialize Data Register D0 to $FF

```
TUTOR 1.3 > .D0 $FF <CR>
```

3. Run the program using the GO (G) Command.

```
TUTOR 1.X > G $2000 <CR>
```

4. TUTOR responds with the following:

```
PHYSICAL ADDDRESS=00002000
1035 000000FF 1039
BUS TRAP ERROR
```
(All 68000 internal registers)
(Next instruction to be executed)

5. The three fields above the expression "BUS TRAP ERROR" indicate where the exception occurred. The left-most field indicates the status of the MC68000 when the exception occurred. Only the five least significant bits of this field are ever modified. For further information, refer to Section 4.3.5 on page 4-30 of the MC68000 Educational Computer Board User's Manual. The middle field indicates the address which caused the fault, and the right-most field is the assembled instruction at which the fault occurred.

6. Describe how and why the Bus Trap Error exception occurred, and at what instruction.

C. Illegal Instruction Exception (Exception Vector No. 4)

1. Enter the following code:

```
TUTOR 1.X > MM $2000;W <CR>

002000   XXXX ? 4AFA <CR>
002002   XXXX ? . <CR>
```

2. Run the program using the GO (G) Command.

```
TUTOR 1.X > G $2000 <CR>
```

3. TUTOR responds with the following:

```
PHYSICAL ADDDRESS=00002000
ILLEGAL INSTRUCTION
```
(All 68000 internal registers)
(Current instruction which caused the exception)

4. Describe why the Illegal Instruction exception occurred. What is the purpose of the $4AFA instruction? List any other opcodes, instructions, etc., which cause this exception to occur.

D. Privilege Violation Exception (Exception Vector No. 8)

1. Enter the following program:

```
TUTOR 1.X > MM $2000;DI <CR>
```

Address	Instruction	Comments
002000	**ANDI.W #$0700,SR <CR>**	To be determined
002004	**002004 BRA $2000 <CR>**	
002006	**. <CR>**	

2. Initialize the Status Register (SR) so that the 68000 is operating in Supervisor Mode.

```
TUTOR 1.X > .SR $FFFF <CR>
```

3. Examine the Status Register using TUTOR's Display Formatted Registers (DF) Command:

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

The Status Register should resemble the following:

```
SR=FFFF=TS7XNZVC
```

4. Run the program using the GO (G) Command.

```
TUTOR 1.X > G $2000 <CR>
```

5. TUTOR responds with the following:

```
PHYSICAL ADDDRESS=00002000
PRIV TRAP ERROR
```
(All 68000 internal registers)
(Current instruction which caused the exception)

6. Describe how and why the Privilege Violation exception occurred. If you don't understand why, trace through the program.

E. Zero Divide Exception (Exception Vector No. 5)

1. Enter the following program:

```
TUTOR 1.X > MM $2000;DI <CR>
```

Address	Instruction	Comments
002000	**DIVU D1,D2 <CR>**	To be determined
002002	**BRA $2002 <CR>**	
002004	**. <CR>**	

2. Initialize Data Registers D1 and D2 so as to cause the Divide By Zero exception to occur.

```
TUTOR 1.X > .D1 $0000 <CR>
TUTOR 1.X > .D2 $1000 <CR>
```

3. Run the program using the GO (G) Command.

```
TUTOR 1.X > G $2000 <CR>
```

4. TUTOR responds with the following:

```
PHYSICAL ADDDRESS=00002000
DIV 0 TRAP ERROR
```
(All 68000 internal registers)
(Next instruction to be executed)

5. Describe how and why the above Zero Divide exception occurred.

6. When performing a division operation and an overflow condition occurs, will exception processing occur? If yes, describe which exception occurs. If no, describe a method for invoking an exception for overflow conditions.

F. Check Instruction Exception (Exception Vector No. 6)

1. Enter the following program:

```
TUTOR 1.X > MM $2000;DI <CR>
```

Address	Instruction	Comments
002000	**CHK.W D6,D7 <CR>**	To be determined
002002	**BRA $2002 <CR>**	
002004	**. <CR>**	

2. Initialize Data Registers D6 and D7 so as to cause the Divide By Zero exception to occur.

```
TUTOR 1.X > .D6 $3000 <CR>
TUTOR 1.X > .D7 $3010 <CR>
```

3. Run the program using the GO (G) Command.

```
TUTOR 1.X > G $2000 <CR>
```

4. TUTOR responds with the following:

```
PHYSICAL ADDDRESS=00002000
CHCK TRAP ERROR
```
(All 68000 internal registers)
(Next instruction to be executed)

5. Describe how and why the CHK Instruction exception occurred. Describe the advantages of the CHK Instruction.

G. Line 1010 Emulator Exception (Exception Vector No. 10)

1. Enter the following code:

```
TUTOR 1.X > MM $2000;W <CR>
002000   XXXX ? A000 <CR>
002002   XXXX ? . <CR>
```

2. Run the program using the GO (G) Command.

```
TUTOR 1.X > G $2000 <CR>
```

3. TUTOR responds with the following:

```
PHYSICAL ADDDRESS=00002000
1010 TRAP ERROR
```
(All 68000 internal registers)
(Current instruction which caused the exception)

4. Describe why the LINE 1010 Emulator exception occurred. What purpose does this exception serve?

H. Line 1111 Emulator Exception (Exception Vector No. 11)

1. Enter the following code:

```
TUTOR 1.X > MM $2000;W <CR>
002000   XXXX ? F000 <CR>
002002   XXXX ? . <CR>
```

2. Run the program using the GO (G) Command.

```
TUTOR 1.X > G $2000 <CR>
```

3. TUTOR responds with the following:

```
PHYSICAL ADDDRESS=00002000
1111 TRAP ERROR
```
(All 68000 internal registers)
(Current instruction which caused the exception)

4. Describe why the LINE 1111 Emulator exception occurred. What purpose does this exception serve? What is this exception specifically intended for?

Discussion:

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

1. An updated version of the programs presented above with both global and local comments included.

2. Answers to each of the questions previously asked.

SAMPLE PROGRAM 3.2 - Bus Error Exception Processing

Purpose:

The purpose of this program is to illustrate how to implement an exception routine.

Given:

- TRAP Function No. 227 outputs a string followed by a <CR> (carriage return) and a <LF> (line feed) to the terminal.

Procedure:

1. Enter the following program by using TUTOR's Memory Modify (MM) Command, and invoking the disassembler.

```
TUTOR 1.X > MM $950;DI <CR>
```

Address	Instruction	Comments
000950	**MOVE.L #$2000,A5 <CR>**	To be determined
000956	**MOVE.L #$201A,A6 <CR>**	
00095C	**MOVE.B #227,D7 <CR>**	
000960	**TRAP #14 <CR>**	
000962	**MOVE.B #228,D7 <CR>**	
000966	**TRAP #14 <CR>**	
000968	**. <CR>**	

2. Use the Memory Set (MS) Command to load a string into memory.

```
TUTOR 1.X > MS $2000 'A BUS ERROR JUST OCCURRED!'<CR>
```

This instruction loads the equivalent ASCII character code of each letter into memory.

3. Verify that the previous instruction was properly executed by invoking TUTOR's Block of Memory Search (BS) Command.

```
TUTOR 1.X >BS $900 $3FFF 'A BUS ERROR JUST OCCURRED!'<CR>
```

If the string was properly stored, TUTOR responds as follows:

```
PHYSICAL ADDRESS=00000900 00003FFF 002000 'A BUS ERROR JUST
OCCURRED!'
```

4. Change the address of the Bus Error Exception Routine that is stored in the 68000's Exception Vector Table.

```
TUTOR 1.X > MM $8 <CR>
000008   00 ? <CR>
000009   00 ? <CR>
```

```
00000A   80 ? 09 <CR>
00000B   30 ? 50 <CR>
00000C   00 ? .<CR>
```

5. Enter the following program by using TUTOR's Memory Modify (MM) Command, and invoking the disassembler.

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

Address	Instruction	Comments
001000	**MOVE.B $50000,D0 <CR>**	To be determined
001006	**BRA $1000 <CR>**	
001008	**. <CR>**	

6. Run the program from Procedure Step #4 using TUTOR's GO (G) command.

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

7. The string 'A BUS ERROR JUST OCCURRED!' should appear on the terminal, followed by the TUTOR prompt.

```
PHYSICAL ADDRESS=00001000
A BUS ERROR JUST OCCURRED!

TUTOR 1.X >
```

8. Depress the RESET Switch.

9. Run the program from Procedure Step #4 again by using the GO (G) command.

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

10. Record any information displayed on the screen. Did anything different occur since the first time you executed the program?

11 Demonstrate to your Lab Instructor that the program from procedure Step #5 is working correctly.

12. Modify the existing program by adding the ability to display on the terminal, the contents of the following 3 registers:

    SSW = Supervisor Status Word
    BA = Bus Address
    IR = Instruction Register

    The contents of these registers should be displayed on a single line underneath the "A BUS ERROR JUST OCCURRED!" message.

13. Run the program from Procedure Step #12, and verify that it is operating properly.

14. Demonstrate to your Lab Instructor the working version of the program from Procedure Step No. 12.

Discussion:

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

1. Add both global and local comments to the programs in Procedures #1, #5 and #12.

2. If you were writing your own Bus Error Exception routine, what type of functions or features would you include in your routine and why?

3. Explain why the string 'A BUS ERROR JUST OCCURRED' didn’t appear on the screen after the program was executed a second time.

SAMPLE PROGRAM 3.3 - The 68000's HALT Mode

Purpose:

The purpose of this exercise is to simulate the conditions necessary to place the 68000 in its HALTED state. This program identifies how the user can tell whether the microprocessor is in a halted state.

Given:

- The starting address of the program that generates a bus error exception is $1000.
- Two consecutive Bus Error exception conditions generate a HALT condition.

Procedure:

1. Generate the first bus error condition by entering the following program into memory using TUTOR's Memory Modify (MM) Command, and invoking the disassembler.

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

Address	Instruction	Comments
001000	**MOVE.B $50000,D0 <CR>**	To be determined
001006	**BRA $1000 <CR>**	
001008	**. <CR>**	

2. Generate the second bus error condition by modifying the Bus Error Exception Vector.

```
TUTOR 1.X > MM $8 <CR>
000008   00 ? <CR>
000009   00 ? FF <CR>
00000A   80 ? FF <CR>
00000B   30 ? FE <CR>
00000C   00 ? . <CR>
```

3. Execute the program using the GO (G) Command.

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

The terminal will display the following:

```
PHYSICAL ADDRESS=00001000
```

Examine the HALT L.E.D on the Front Panel of the SANPER-1 ELU. Record its status (ON or OFF)

Alternatively, a HALT LED is also provided on the SANPER-1 System Board, and is labeled HALT or LED1. The LED is located near the front middle of the printed circuit board.

4. Depress the ABORT Switch (the one with the RED cap) on the front panel of the SANPER-1 Educational Lab Unit. Record what events occur once this switch is depressed. Specifically, record the state of the HALT LED, and any other LEDs, and any information that appears on the terminal.

5. Depress the RESET Switch (the one with the BLACK cap) on the front panel of the SANPER-1 Educational Lab Unit. Record what events occur once this switch is depressed. Specifically, record the state of the HALT LED, and any other LEDs, and any information that appears on the terminal.

Discussion:

Submit the following to your Lab Instructor:

1. A commented version of the program outlined in Procedure Step No. 1 (include both global and local comments).

2. With regard to this sample program, describe in detail the sequence of events, which caused the 68000 to enter its HALTED state.

3. In general, what sequences of events causes a double bus fault to occur?

4. Describe what effect a double bus error condition has on the 68000's HALT signal. Discuss the advantages and disadvantages of this feature.

5. Describe the result of depressing the ABORT switch and explain the reason for this.

6. Describe the result of depressing the RESET switch and explain the reason for this.

7. Explain the differences between the RESET and ABORT switches. Under what conditions would you use the ABORT switch? When you reset the lab unit, what happens to the contents of the Exception Vector Table?

8. What are the differences between manually activating the RESET pushbutton, and having the 68000 execute the RESET instruction?