Experiment No. 03:

Exception Processing and System Control

By: Theo Guidroz

Lab Partner: Daniyal Khan, Gregory Bonnema

Instructor: Dr. Jafar Saniie

ECE 441-001

Lab Date: 02-05-2020

Due Date: 02-12-2020

Acknowledgment: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced.

Signature : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**I. Introduction**

**A. Purpose**

The purpose of this lab is to help students understand how M68000 microprocessor handles exceptions. Students will program code to throw errors and they will have to recognize the exceptions and what is causing them.

**B. Background**

An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions. When such an event occurs, the status register is set to exception processing and a subroutine for the specific exception is thrown to handle the situation. The M6800 microprocessor possesses an exception table which contains the address of exception handlers.

**II. Lab Procedure and Equipment List**

**A. Equipment**

*Equipment*

* SANPER-1 system
* PC with TUTOR software

**B. Procedure**

Sample Program 3.1

Address Error Exception

1. Type program 3.1A into EASy68k and run it.
2. Use the LO1 command and the “send ASCII” tab to transfer and program the code to SANPER-01.
3. Initialize D0 to $FF.
4. Run the program.
5. Observe the exception thrown.

Bus Error Exception

1. Type program 3.1B into EASy68k and run it.
2. Use the LO1 command and the “send ASCII” tab to transfer and program the code to SANPER-01.
3. Initialize D0 to $FF.
4. Run the program.
5. Observe the exception thrown.

Illegal Instruction Exception

1. Type program 3.1C into EASy68k and run it.
2. Use the LO1 command and the “send ASCII” tab to transfer and program the code to SANPER-01.
3. Run the program.
4. Observe the exception thrown.

Privilege Violation Exception

1. Using the MM command, enter in program 3.1 D.
2. Initialize the status register to $FFFF.
3. Examine the status register using the DF command.
4. Run the program.
5. Observe the exception thrown.

Zero Divide Exception

1. Type program 3.1E into EASy68k and run it.
2. Use the LO1 command and the “send ASCII” tab to transfer and program the code to SANPER-01.
3. Initialize D1 to $0000 and D2 to $1000.
4. Run the program.
5. Observe the exception thrown.

Check Instruction Exception

1. Type program 3.1F into EASy68k and run it.
2. Use the LO1 command and the “send ASCII” tab to transfer and program the code to SANPER-01.
3. Initialize D6 to $3000 and D7 to $3010.
4. Run the program.
5. Observe the exception thrown.

Line 1010 Emulator Exception

1. Type program 3.1G into EASy68k and run it.
2. Use the LO1 command and the “send ASCII” tab to transfer and program the code to SANPER-01.
3. Run the program.
4. Observe the exception thrown.

Line 1111 Emulator Exception

1. Type program 3.1H into EASy68k and run it.
2. Use the LO1 command and the “send ASCII” tab to transfer and program the code to SANPER-01.
3. Run the program.
4. Observe the exception thrown.

Sample Program 3.2

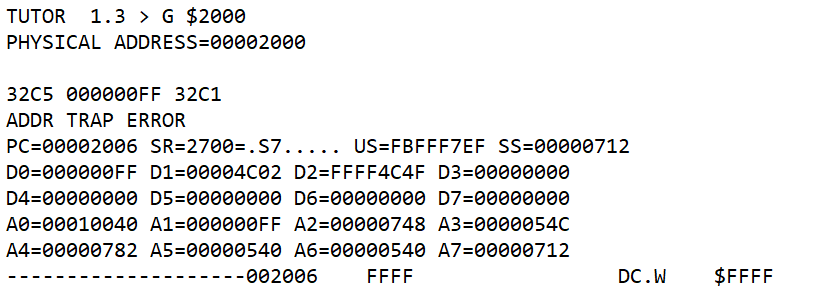
1. Type program 3.2 into EASy68k and run it.
2. Use the LO1 command and the “send ASCII” tab to transfer and program the code to SANPER-01.
3. Use the MS command to enter in the string ‘A BUS ERROR JUST OCCURRED!’ at $2000.
4. Use MM to change the address of the Bus Error Exception Routine using.
5. Type program 3.2 at step 5 into EASy68k and run it.
6. Use the LO1 command and the “send ASCII” tab to transfer and program the code to SANPER-01.
7. Run the program from step 4.
8. Run the program again.
9. Observe the output and record any differences.
10. Type program 3.2 at step 12 into EASy68k and run it.
11. Run the program from step 9 and verify that the SSW, BA, and IR are displayed.

Sample Program 3.3

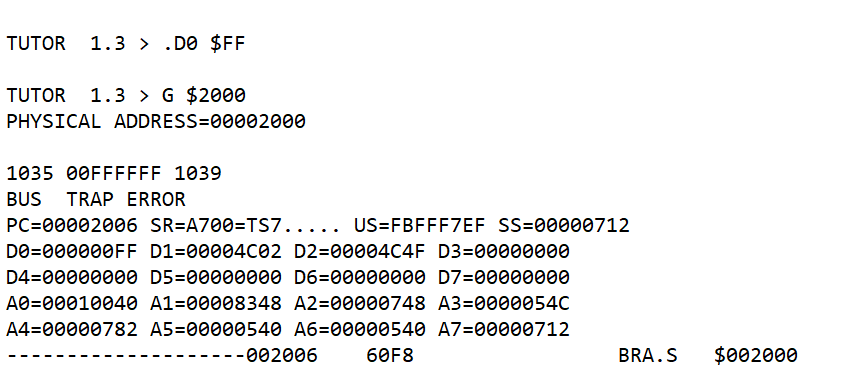
1. Type program 3.3 into EASy68k and run it.
2. Use the LO1 command and the “send ASCII” tab to transfer and program the code to SANPER-01.
3. Modify the Bus Error Exception Vector using MM and run the program.
4. Check the HALT LED on the front panel of the SANPER-1 ELU.
5. Depress the abort switch and record the state of the HALT LED and any other LEDs.
6. Depress the reset switch and record the state of the HALT LED and any other LEDs.

**III. Results and Analysis**

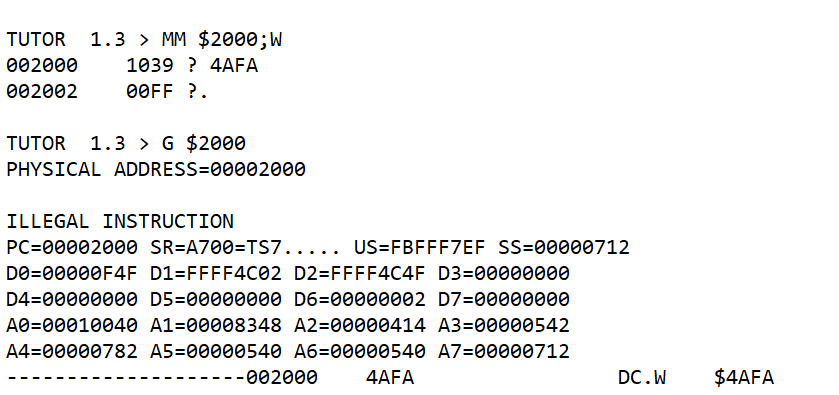
Sample Program 3.1



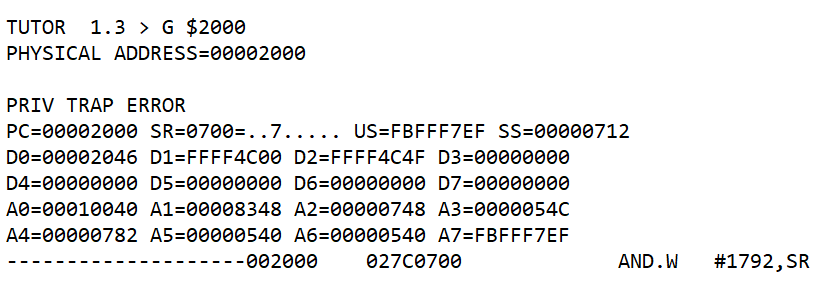
**Figure 1: Result of Address Error Exception Sample Program**



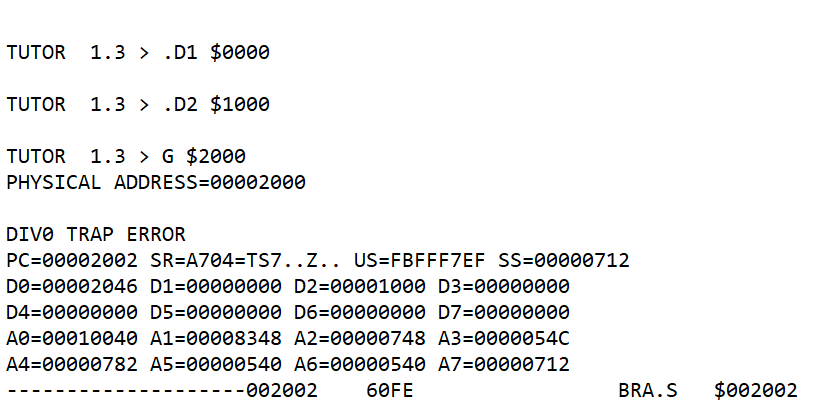
**Figure 2: Result of Bus Error Exception Sample Program**



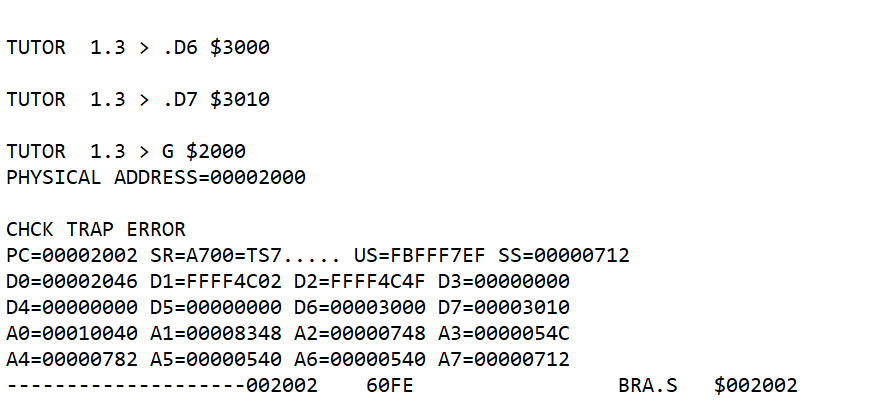
**Figure 3: Result of Illegal Instruction Exception Sample Program**



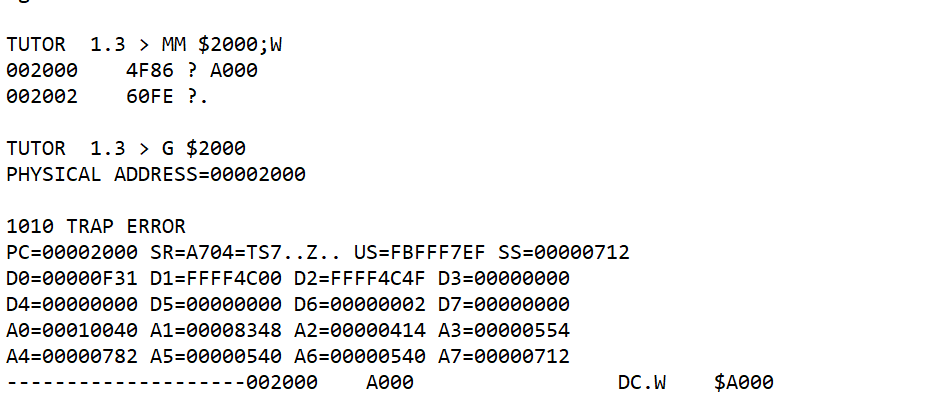
**Figure 4: Result of Privilege Exception Sample Program**



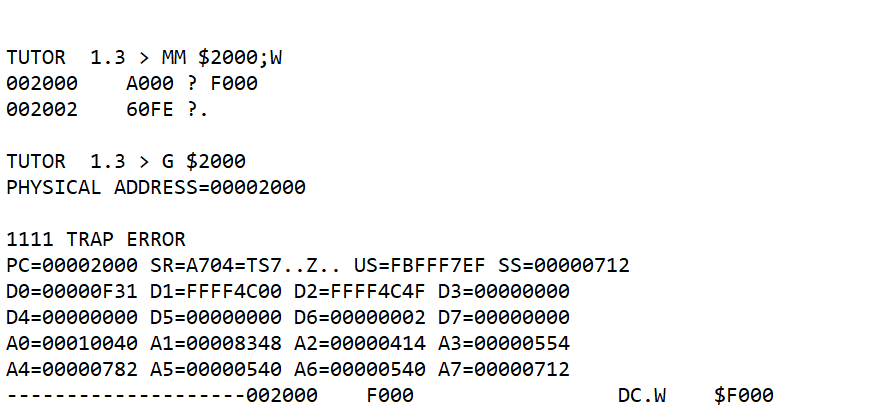
**Figure 5: Result of Divide by 0 Exception Sample Program**



**Figure 6: Result of Check Error Exception Sample Program**



**Figure 7: Result of 1010 Error Exception Sample Program**



**Figure 8: Result of 1111 Error Exception Sample Program**

Sample Program 3.2

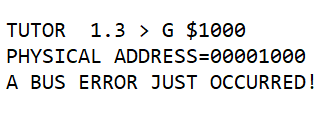


Figure 9: Result of Bus Error Exception Processing Sample Program

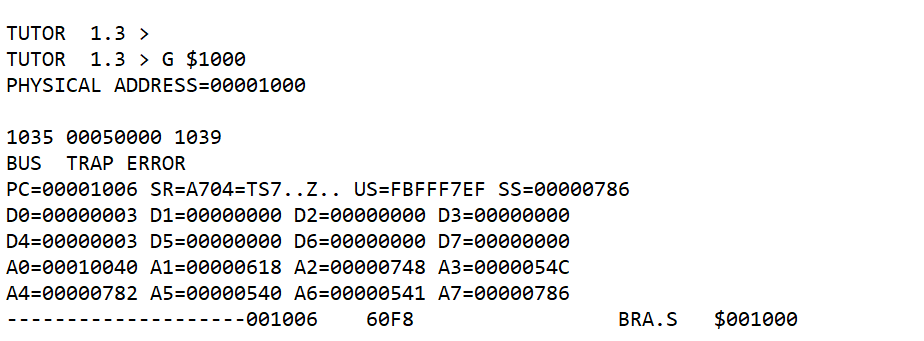


Figure 10: Result of Bus Error Exception Processing Sample Program when run a second time

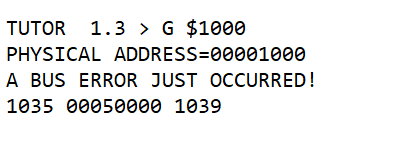


Figure 11: Result of Bus Error Exception Processing Sample Program with printed SSW, BA, IR

Sample Program 3.3

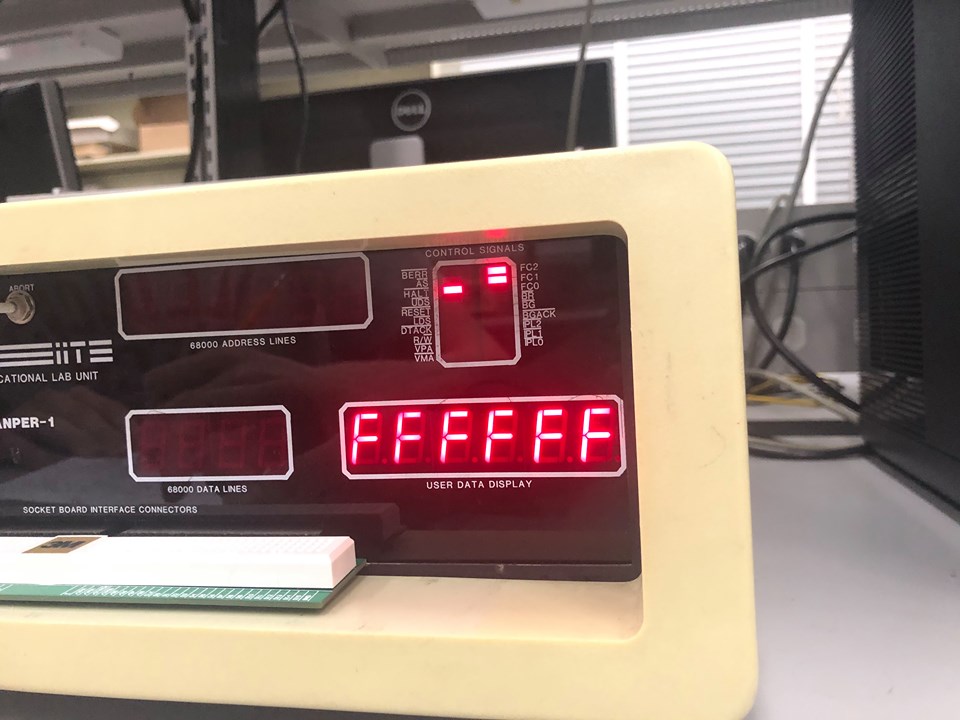


Figure 12: Results of Sample Program 2.3 (Depressing ABORT)

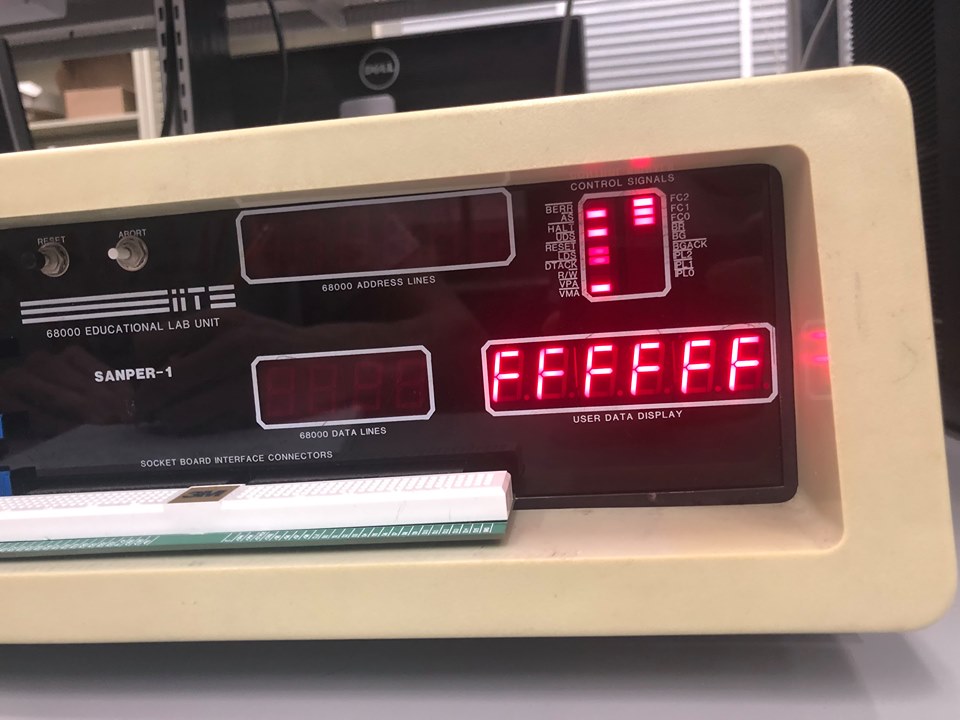


Figure 13: Results of Sample Program 2.3 (Depressing RESET)

1. **Discussion**

Programs

MOVE.W D0,A1 ; Move word from D0 to A1

MOVE.W D1,(A1)+ ; Move word from D1 into memory address pointed to by A1 and increment A1 by 2

BRA $2000 ; Branch to $2000

Figure 1: Sample Program 3.1 A

MOVE.B $FFFFFF,D0 ; Move value from memory address $FFFFFF into D0

BRA $2000 ; Branch to $2000

Figure 2: Sample Program 3.1 B

ANDI.W #$0700,SR ; AND $700 with contents of status reg

BRA $2000 ; Branch to $2000

Figure 3: Sample Program 3.1 D

DIVU D1,D2 ; Divide D2 by D1

BRA $2002 ; Branch to $2002

Figure 4: Sample Program 3.1 E

CHK.W D6,D7 ; Compare D7 to D6

BRA $2002 ; Branch to $2002

Figure 5: Sample Program 3.1 F

MOVE.L #$2000,A5 ; Move address $2000 to A5

MOVE.L #$201A,A6 ; Move address $201A to A6

MOVE.B #227,D7 ; Syscall for printing a string to terminal

TRAP #14 ; TRAP function

MOVE.W (SP)+,D0 ; Move SSW to D0

MOVE.W #232,D7 ; Syscall to convert 4 hex digits (word) to ASCII

TRAP #14 ; TRAP function

MOVE.B #$20,(A6)+ ; Hex for space character

MOVE.L (SP)+,D0 ; Move BA from SSP to D0

MOVE.W #230,D7 ; Syscall to convert 8 hex digits (long) to ASCII

TRAP #14 ; TRAP function

MOVE.B #$20,(A6)+ ; Adding a space character

MOVE.W (SP)+,D0 ; Move IR from SSP to D0

MOVE.W #232,D7 ; Syscall to convert 4 hex digits (word) to ASCII

TRAP #14 ; TRAP function

MOVE.W #227,D7 ; Syscall to print registers

TRAP #14 ; Trap function

MOVE.W #228,D7 ; go to TUTOR

TRAP #14 ; Trap function

Figure 6: Sample Program 3.2 (Procedure 12)

MOVE.B $50000,D0 ; Move byte at $50000 to D0

BRA $1000 ; Branch to $1000

Figure 7: Sample Program 3.2 (Procedure 5)

MOVE.B $50000,D0 ; Move byte at $50000 to D0

BRA $1000 ; Branch to $1000

Figure 8: Sample Program 3.3

Questions

Sample Program 3.1

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

Discussion section.

1. **3.A: Describe how and why the above Address Trap exception occurred.**

D0 was initialized to $FF and was copied into A1. Since $FF is odd, a word cannot be moved at the address location specified by A1 ($FF). Since the word form D1 is being moved to an odd address, an address trap error occurred.

1. **3.B: Describe how and why the Bus Trap exception occurred, and at what instruction.**

The error occurred because the instruction at $2000 tries to access data in memory at location $FFFFF but this location is out of range.

1. **3.C: Describe why the Illegal Instruction exception occurred. What is the purpose of the $4AFA instruction? List any other opcodes, instructions, etc. Which caused this exception to occur?**

The instruction $4AFA does not exist since its opcode does not map to any implemented instructions. Therefore, the exception was thrown.

1. **3.D: Describe how and why the Privilege Violation exception occurred. If you don’t understand why, trace through the program.**

The Privilege Violation exception occurred because the program, which is ran in user mode on the second iteration, tries to modify the status register and this can only be done in supervisor mode.

1. **3.E: When performing a division operating 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.**

No exception occurs. To invoke an exception for this case, a subroutine, who throws an exception when the V bit in the SR is set, has to be created.

1. **3.F: Describe how and why the CHK Instruction exception occurred. Describe the advantages of the CHK instruction.**

The CHK Instruction occurs if the data checked is less than zero or greater than the first operand on the line code. Since the value of D7 is initialized to be greater than the value of D6, the exception occurred. CHK instruction is useful in array processing. This instruction performs the bound check which compares an array index against zero and the limit value addressed by the instruction. If the index is out of bounds, a trap will occur.

1. **3.G: Describe why the LINE 1010 Emulator exception occurred. What purpose does this exception serve?**

Any instruction whose opcode starts with ‘1010’ in base 2 or A in hex value invokes the exception. The purpose of this exception is to allow users to customize an instruction as an exception handler.

1. **3.H: Describe why the LINE 1111 Emulator exception occurred. What purpose does this exception serve? What is this exception specifically intended for?**

Any instruction whose opcode starts with ‘1111’ in base 2 or F in hex value invokes the exception. The purpose of this exception is to allow users to customize an instruction as an exception handler.

Sample Program 3.2

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

Discussion section.

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

I would use the CHK instruction with the first operand being the immediate value of the upper limit of accessible range in memory. Then the routine would display a message and recall the user of the accessible range in memory.

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

The bus error exception routine has been modified to print out the string. When the program was reset, the string was no more included in the routine and therefore, it was not printed when the program was executed the second time.

Sample Program 3.3

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

Discussion section.

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

At address $1000, the program attempts to move data from out of range memory in D0. This instruction throws a bus trap exception. The routine has been modified to point to address $FFFFE, which is also out of range. Since a double bus error occurred, the process is sent into a HALTED state.

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

Two exceptions occurring on the same line will prevent each other from being processed and hence cause double bus fault.

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

The double bus error condition asserts the 68000’s HALT signal. The advantage of this feature is it saves the system from damage. The disadvantage is that HALT will not reboot your system; the system must be manually rebooted.

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

The ABORT switch exits the current program but since the microprocessor was in HALTED mode, it had no control and stayed in that state

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

The RESET resets the vector tables, PC, and SSP. Since the microprocessor if fully reset, it is no more in HALTED mode and can function properly.

1. **Explain the difference 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?**

The RESET resets the vector tables, PC, and SSP to their initial values, and restores the microprocessor to its default state. The ABORT switch exits the executing program but does not affect the exception vector table. The ABORT switch is used to get out of an infinite loop.

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

The RESET pushbutton reset all externa devices as well as internal components such as register and SSP whereas the RESET instruction only reset internal components.

**IV. Conclusion**

Students became familiar with interrupts and exceptions handling by the end of this laboratory. These concepts are essential for developing more complex programs using microprocessors. Knowledge of error handling will definitely come in handy in the future labs.

**References**

[1] Experiment 3 Lab Manual

[2] Educational Computer Board manual appendix

**Attachments**