Experiment No. 3:

EXCEPTION PROCESSING AND SYSTEM CONTOL

By: Haomin Shi

Lab Partner: Anh Nguyen

Instructor: Dr. Jafar Saniie

ECE 441-003

Lab Date: 09-21-2017

Due Date: 09-28-2017

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 let students become familiarized with the MC68k exception processing, the TUTOR exception handling, and the MC68k system controls

B. Background

This lab allowed students to run the sample program provided by the lab manual in order to observe the MC68k's system controls and exception handling, also with the TUTOR's exception handling.

II. Lab Procedure and Equipment List

A. Equipment

Equipment

- SANPER-1 system
- PC with TUTOR software

B. Procedure

1. The student will follow the lab instructions and input the program into the SANPER unit, record the data and observe the system when needed.

III. Results and Analysis

A. Sample Program 3.1.A

The source code of program 3.1.A:

```
ORG $2000
    START:
        MOVE.W D0, A1       ;LOAD D0 TO ADDRESS
        MOVE.W D1, (A1)+    ;INCREAING A1 TO ODD, CAUSE PROBELM
        BRA $2000           ;
    END START
```

The reason there is an address trap occur is that the processor is trying to access a word or long-word operand or an instruction that is at an odd address. For example, in this sample, the D0 is set to FF, which is an odd number, thus, the error occurred.

B. Sample Program 3.1.B

The source code of program 3.1.B:

```
ORG $2000
    START:
        MOVE.B $FFFFFF, D0  ;TRY LOAD A CRAZY ADDRESS
        BRA $2000           ;
    END START
```

The trap appeared, since the number we load into the register is too big, the CPU could not access that memory address.

C. Sample Program 3.1.C:

Based on the M68K user manual, these patterns are causing illegal instruction trap because on MC68k, special patterns are reserved for Motorola and customer use.

D. Sample Program 3.1.D:

The source code of program 3.1.D:

```
ORG $2000
    START:
        ANDI.W #$0700, SR  ;TRY AND SR
        BRA $2000        ;
    END START
```

The reason this error appears is because of that, ANDI SR the CPU need to be in SV mode. However, when we are running this instruction, we were doing it in User mode, thus the privilege violation occurred.

E. Sample Program 3.1.E

The source code of program 3.1.E:

```
ORG $2000
    START:
        DIVU D1, D2     ;DIVID BY 0
        BRA $2002       ;
    END START
```

This exception is easy to explain, DIVU will cause an exception if it’s dividing number “0”. Here in D1, it contains 0, thus the error occurred.

The source of code program 2.5:

F. Sample Program 3.1.F

The source code of program 3.1.F:

```
ORG $2000
    START:
        CHK.W D6, D7    ;CHECK D7 < D6
        BRA $2002
    END START
```

The exception occurs here if the data register is less than zero or greater than the upper bound contained in the operand.

G & H. Sample Program 3.1.G and 3.1.H

According to the M68K user manual, the 1010 and 1111 are designed to be reserved for Motorola, thus we had an emulator exception here.

J. Sample Program 3.2

a. Program for Procedure 1:

```
ORG $950
    START:
        MOVE.L #$2000, A5
        MOVE.L #$201A, A6
        MOVE.B #227, D7
        TRAP #14
        MOVE.B #228, D7
        TRAP #14
    END START
```

b. Program for Procedure 5:

```
ORG $1000
    START:
       MOVE.B $50000, D0
       BRA $1000
    END START
```

c. Program for Procedure 12:

```
ORG $950
    START:
       MOVE.L #$2000, A5      ;
       MOVE.L #$201A, A6      ;
       MOVE.B #227, D7        ;PRINT STR
       TRAP #14
       MOVE.W (A7)+, D0       ;MOVE SSW
       MOVE.W #232, D7        ;ASCII -> HEX
       TRAP #14
       MOVE.B #$20, (A6)+     ;SPACE
       MOVE.L (A7)+, D0       ;MOVE BA
       MOVE.W #230, D7        ;
       TRAP #14
       MOVE.B #$20, (A6)+     ;SPACE
       MOVE.W (A7)+, D0       ;MOVE IR
       MOVE.W #232, D7        ;
       TRAP #14               ;
       MOVE.B #$20, (A6)+     ;SPACE
       MOVE.W #227, D7        ;PRINTLN
       TRAP #14               ;
       MOVE.W #228, D7        ;BACK TO TUTOR
       TRAP #14               ;

    END START
```

Bus error routine made debugging process easier, it helped us to pinpoint the error. The reason why the bus error still appeared on the screen after the program was executed a second time is that the reset only reset the bus error vector, but not the address of the routine.

K. Sample Program 3.3

Photos:

Photos are in the appendix section.

Procedure 1:

```
ORG $1000
    START:
       MOVE.B $50000, D0    ;CAUSE BUS ERROR
       BRA $1000
    END START
```

The reason processor is halted is because we changed the address that is supposed to point to the bus error routine, it's no longing pointing towards the right address, thus causing the error.

In general, processor halted because of faulty error handling code, or hardware problem.

The double bus fault stops the executing of CPU. However, it can prevent further damage to the hardware.

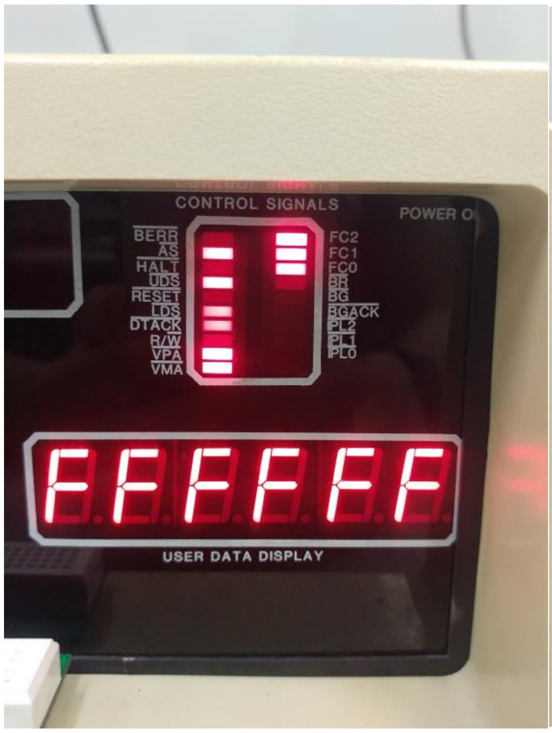
The abort stops the current running code and returns to TUTOR. The reset button aborts the code and resets the processor + the vector table.

The RESET instruction is privileged and it is an instruction. It is an instruction used to perform a software controlled reset for all peripherals connected to the MC68k.

IV. Conclusions

At the end of this lab, students became familiarized with the exceptions and its handling. The lab is accomplished.

Appendix



References

[1] Experiment 3 Lab Manual

[2] Educational Computer Board manual appendix