Lab 08 report: Exceptions and I/O

Objectives

- To understand the exception mechanism in MIPS
- To understand Coprocessor 0 instructions
- To write exception handlers
- To understand Memory Mapped I/O

Introduction

Branches and jumps provide ways to change the control flow in a program. Exceptions can also change the control flow in a program. The MIPS convention calls an exception any unexpected change in control flow regardless of its source (i.e. without distinguishing between a within the processor source and an external source).

An exception is said to be synchronous if it occurs at the same place every time a program is executed with the same data and the same memory allocation. Arithmetic overflows, undefined instructions, page faults are some examples of synchronous exceptions. Asynchronous exceptions, on the other hand, happen with no temporal relation to the program being executed. I/O requests, memory errors, power supply failure are examples of asynchronous events.

An interrupt is an asynchronous exception. Synchronous exceptions, resulting directly from the execution of the program, are called traps.

When an exception happens, the control is transferred to a different program named exception handler, written explicitly for the purpose of dealing with exceptions. After the exception, the control is returned to the program that was executing when the exception occurred: that program then continues as if nothing happened. An exception appears as if a procedure (with no parameters and no return value) has been inserted in the program.

Tasks

Task1:

1. **Requirement:** it is required to implement a MIPS program that prompts the user to enter two integers x and y then divide x by y. However, if the user entered 0 in y then the program will raise an exception to inform the user modify the error and offer him to modify it to perform the division.
2. **Approach:** we have to prompt the user to enter the integer x then save its value to use it later. Then we have to prompt him to enter the value of y. However, if the user entered 0 in y then there is a trap instruction that will compare y with 0 to raise an exception if they are equivalent. When the exception is raised the program will handle the exception by identifying the type of the exception by MIPS exception key. After the program have identified the exception type as a trap exception, it will identify the type of the trap exception key that is written by the developer to print the suitable exception message *"Divide by Zero Exception. Please enter a different value for y."* then return to the prompt instruction to ask the user to enter number y again (normal flow of the program). If no exception occurred the program will divide x by y then display the result.

Tasks

Task2:

1. **Requirement:** it is required to implement a MIPS program that reads every alphabet character that user enters and switch its case (it will not modify any character that is not alphabet). This is required for every character that is typed on the keyboard.
2. **Approach:** to implement we first have to wait the user to click any character key on the keyboard. When the user clicks on the keyboard the program well takes the character entered and store it in a register. The program then will check if the character entered is an alphabet according to the ASCII table by the following:
    - **a.** If the code of the letter is less than 'A' then it is not an alphabet letter, then the program will go to the display instructions.
    - **b.** If the code of the letter is greater than 'z' then it is not an alphabet letter, t then the program will go to the display instructions.
    - **c.** If the code of the letter is greater than 'Z' then it may be a lower-case alphabet character. If it is not it is an upper-case character then the program will switch its case, then the program will go to the display instructions.
    - **d.** If it has been decided to be a lower-case character but first it should be greater than 'a'. If it is less than 'a' then it is non-alphabetic character (between 'Z' and 'a') then then the program will go to the display instructions.
    - **e.** If the code of the letter is greater than 'a' then the program will switch its case, then the program will go to the display instructions.

    After the program has switched the alphabet letter case it will take the character from the register to display it to the user. This method will occur on every click on the keyboard by looping until the user stop the program.

Conclusion

In conclusion, an exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions. It is useful because there are types of errors that cannot be handled by branches and jumps instructions. Therefore, implementing exception handlers will improve the user experience and prevent the program from crashing.