Topic: Exception Handling

Aimed for: Achieving Robustness and Flexibility *Author: Yara Altamimi*

Objective

To design an efficient exception handler for the MIPS pipelined processor, ensuring the system operates robustly and flexibly in the face of unexpected events.

What is Exception Handling?

Exception handling is the process of detecting, responding to, and recovering from unexpected events that occur during program execution. Exceptions can arise from various causes, such as:

- Arithmetic errors (e.g. division by zero).
- Memory access violations (e.g. accessing an invalid address).
- Illegal instructions (e.g. wrong opcode).
- Arithmetic overflow (e.g. the result is too large or too small to hold in the result register).

Exception Handling Workflow

1. **Detection**
    - Exceptions are detected during various pipeline stages, such as instruction decode (ID) or memory access (MEM), based on the type of error (e.g., invalid instruction, memory violation).
2. **State Management:**
    - **EPC (Exception Program Counter)**: Saves the address of the instruction that caused the exception.
    - **Cause Register**: Records the specific reason or type of the exception for later use by the exception handler.
3. **Pipeline Control:**
    - **Flush**: Clear all partially executed instructions currently in the pipeline to prevent erroneous or incomplete operations.
    - **Redirect**: Modify the program counter (PC) to transfer control to the exception handler's address (if a handler is implemented).

4. **Exception Handling Routine:**
    - Executes predefined routines to resolve or address the exception (if a handler is implemented).
    - **Halted all the operations:** An alternative solution is to halt the whole program and set all the control signals to zero (if an exception is detected, stop the execution).
5. **Resume Execution:**
    - Restore the pipeline state and resume execution, either restarting the faulting instruction or proceeding to the next one (if a handler is implemented).

Benefits

- **Ensures Correct Program Execution**:

  Handles unexpected events, ensuring the program continues running correctly without crashes or data corruption.

- **Maintains System Reliability**:

  Provides a robust mechanism for managing errors and interruptions, ensuring stable and predictable system behavior.

- **Improved Debugging:**

  Provides detailed information about exceptions (through the Cause Register and EPC), helping identify and fix issues quickly, and enhancing the debugging process.

References

For more details, check this reference: Exception and Interrupt handling in the MIPS architecture