



King Fahd University of Petroleum and Minerals  
College of Computer Sciences and Engineering  
Computer Engineering Department  
COE 301: Computer Architecture

# **LAB 08: MIPS Exceptions and I/O**

Saleh AlSaleh



# Agenda

- Exceptions
- Coprocessor 0
- Exception Codes
- Exception Handler
- Memory Mapped I/O
- Live Examples
- Tasks

# Exceptions

- Exception is any unexpected change of control flow, regardless of its source.
- MIPS CPU operates either in the user mode or kernel mode.
- User programs (applications) run in user mode.
- The CPU enters the kernel mode when an exception happens/occurs.
- In MIPS, all the instructions inside text segment are in the try block.

# Coprocessor 0

- Coprocessor 0 has several important registers such as:
  - **Vaddr (\$8)**: Contains the invalid memory address caused by load, store, or fetch.
  - **Status (\$12)**: Contains the interrupt mask and enable bits (see below).
  - **Cause (\$13)**: Contains the type of exception and any pending bits (see below).
  - **EPC (\$14)**: Contains the address of the instruction when the exception occurred.

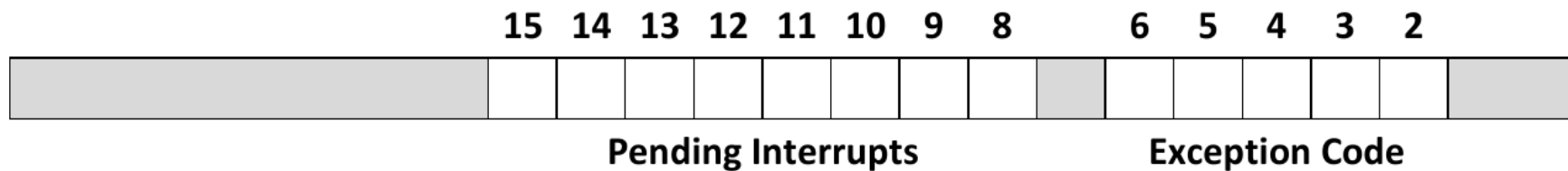


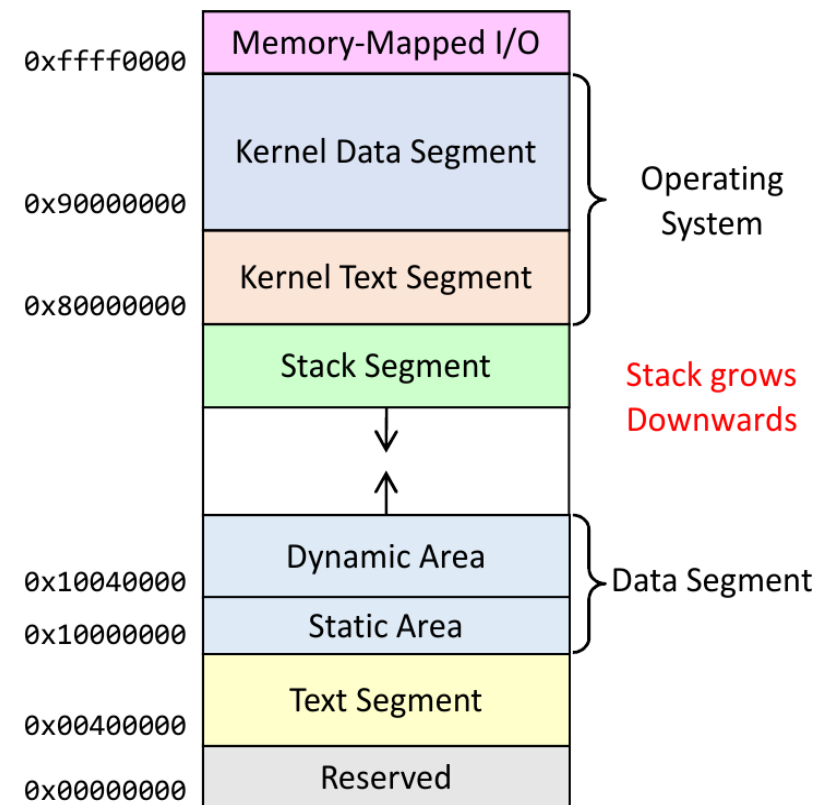
Figure 8.3: The **Cause** Register **\$13**

# Exception Codes

Code	Name	Description
0	INT	Hardware Interrupt
4	ADDRL	Address error exception caused by load or instruction fetch
5	ADDRS	Address error exception caused by store
6	IBUS	Bus error on instruction fetch
7	DBUS	Bus error on data load or store
8	SYSCALL	System call exception caused by the <b>syscall</b> instruction
9	BKPT	Breakpoint exception caused by the <b>break</b> instruction
10	RI	Reserved instruction exception
12	OVF	Arithmetic overflow exception
13	TRAP	Exception caused by a trap instruction
15	FPE	Floating-Point exception cause by a floating-point instruction

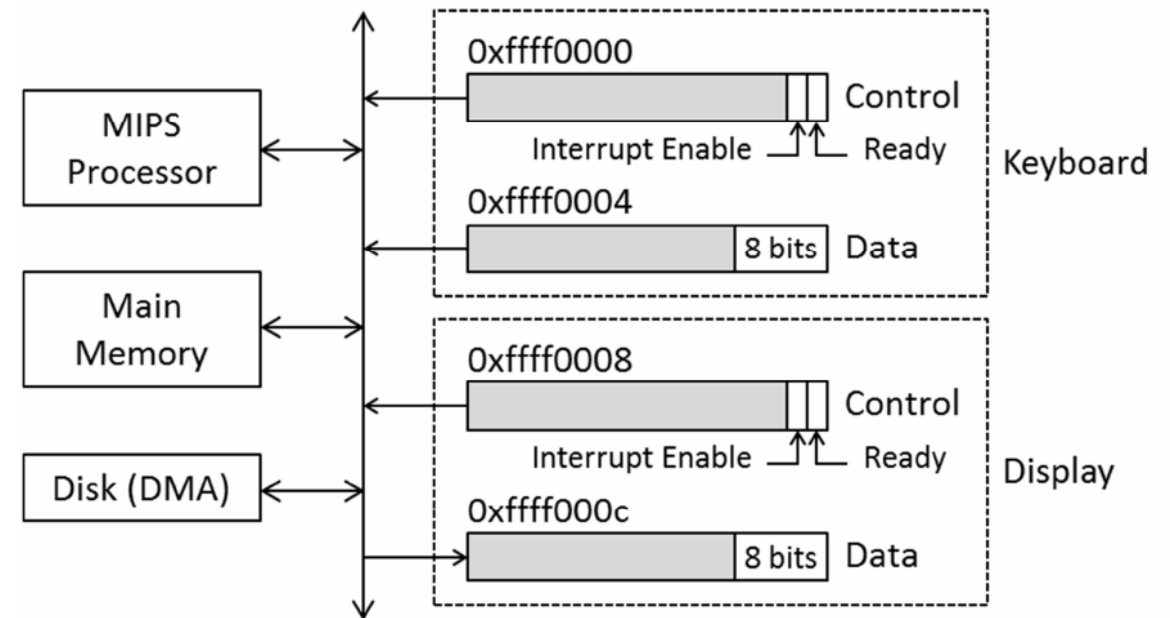
# Exception Handler

- A special piece of code in the kernel text (at address 0x80000180) that handles exceptions.
- There is a default exception handler in MARS. However, it can be overwritten.



# Memory Mapped I/O

- Input/Output devices reside outside the processor chip.
- There are two general ways for the processor to communicate with the I/O devices:
  - Using specialized instruction to communicate with each device.
  - Map I/O device registers to memory space. Then, use load and store instructions to read and write to the devices respectfully.



# Live Examples

# Task #1

- Write a MIPS assembly program that reads two integers from the user  $x$  &  $y$ . If  $y$  is zero, raise an exception and the user should be prompted to enter a different value of  $y$ . If  $y$  is not zero, perform the operation  $x/y$ .
- (Hint: use trap instruction after reading  $y$ )

Sample Run of the task

Enter Dividend (x): 10

Enter Divisor (y): 0

Divide By Zero Exception. Please enter a different value for y.

Enter Divisor (y): 2

The result of  $x / y$  is 5

# Task #2

- Write a MIPS assembly program that reads a string *str* (one character at a time) from the user using Memory Mapped I/O (Do not use syscall). Loop over each character and flip its case (i.e. the uppercase should be small case and vice versa). Finally, print the modified String (one character at a time) again using Memory Mapped I/O (Do not use syscall).

Sample Run of the task

Enter String: **Hello,**  
**World!**  
The string after flipping the  
cases is: **hELLO, wORLD!**