Lab 2: Processor Exceptions

CMPT 328 Computer Architecture

Scott Overholser

This lab was written for the Westminster College CMPT 328 Computer Architecture course Spring 2015 semester.

Last Modified: Tue Feb 17 17:29:20 2015 -0700

Table of Contents

1 Introduction

Lab 2 will build on your familiarity with the development environment and first experience programming the LaunchPad in Lab 1. It will expand the complexity and functionality of the programming project.

2 Lab Objective

In this lab you will learn what processor exceptions are and how they are handled. The scope will include the following:

- How the processor handles fault conditions.
- How the processor manages the stack.
- Understanding vector table in more detail than in Lab 1.
- Understanding context switching.
- Introduce clock gating.

You will also understand that the data sheet is an indispensable resource for understanding the functionality of the LaunchPad.

3 Notes

At this point you should be getting familiar with your development environment. This should mean less technical support problems and more focus on the experiments.

Be sure to connect your USB cable to the debug port on the LaunchPad. Also confirm that the switch is in the debug position.

Take notes of your results and observations as you proceed. These notes should be used as you write your lab report.

3.1 Reference Material

Tiva TM4C123GH6PM Microcontroller Data Sheet will be referenced often in this and future labs. It is in the Canvas Files area filed under Tiva C Series Microcontroller. The filename is `tm4c123gh6pm.pdf`. Please download this file to your computer.

Refer to the following in the TM4C data sheet.

- *2.3.2 Stacks* on page 74.
- *2.5.4 Vector Table* on page 107.
- *2.5.7.1 Exception Entry* on page 109.
- *Register 60: General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO), offset 0x608* on page 340.

4 Experiments

Download the three lab 2 zip files in the Canvas files area from the Labs folder.

- Lab 2 Project Part 1.
- Lab 2 Project Part 2.
- Lab 2 Project Part 3.

They are zip files that will extract into their own directories. Each is an IAR Embedded Workbench project. They have only minor differences. Choose a location and extract the projects.

Review the code. It is well commented. Notice the familiar parts from Lab 1. You should recognize the first two items in the vector table from the project in lab 1. The stack pointer is now set just a bit differently but it is still the same value.

Notice the new entries in the vector table. Refer to the data sheet for a list of processor exceptions. Each of the symbols is resolved to a memory location by the linker at compile time. Each one of the fault handlers is attached to a single instruction `B .` that loops forever.

4.1 Generic Fault Handler

Start the IAR Embedded Workbench by double-clicking on the `workspace.eww` file. Build the executable and then download and debug. Execution will stop at the location of `main`.

Carefully review the data sheet *2.5.7.1 Exception Entry* section to familiarize yourself with the process of context switching on interrupt handling. Note the contents of your registers in the register view.

Single step through the code until the processor exception is triggered. Step into the fault handler. Again note the contents of your registers in the register view. In particular, note the value of the `SP` register.

In the memory view, click into the *Go To* field and enter the value of the `SP` register. This is known as the top of the stack. Compare the contents of the stack in RAM with the register contents of your registers you noted prior to the processor exception being triggered. *Report what you learned here in your lab report. Include your observations.*

Read the data sheet *2.3.2 Stacks* section to understand how the processor uses the stack.

4.2 Create HardFault Handler

Switch to the Lab 2-02 project. Compile the project. Then download and debug. Execution will stop at `main`. Leave it there for now.

Carefully review *2.5.4 Vector Table* on page 107 of the data sheet. In the disassembly view, scroll up to the beginning of flash memory. You will be looking for the vector table. The IAR Embedded Workbench will helpfully have put each of the labels next to the value in each 4-byte location. Make a note of each 4-byte value in the vector table that corresponds to the vector table in your source code. You will begin with the Stack Pointer at `0x0000`, Reset at `0x0004`, etc... *Include this table in your lab report.*

Note at this point that each of the entries in the vector table that correspond to fault handlers are pointing to the same location in memory. We will change that. Stop debugging.

To illustrate how the processor handles interrupts using the vector table, break out `HardFault_Handler` from the other fault handlers that are all grouped together. The following diff shows how it is done:

Rebuild your code. Then download and debug. Single-step through the code in the disassembly window until the hard fault is once again triggered. The new `HardFault_Handler` code should now be executing. *Does this confirm the function of the vector table?*

Stop debugging. Close IAR Embedded Workbench.

4.3 Run Mode Clock Gating

Switch to the lab 2-03 project. Compile the project. Then download and debug. Execution will stop at `main`. Leave it there for now.

Carefully review the new code. Note the instruction that is commented with "Replace this with the correct value." You will be reading the data sheet and determining the value to use in that `MOVS` operand.

Study the RCGCGPIO register description On page 340 of the TM4C data sheet. This describes how to enable GPIO Port F. Once you understand how to program the RCGCGPIO register to enable GPIO Port F, return to your program and enter that value in place of `0x00` on the line referenced above. *Be sure to include a summary of this process in your lab report.*

Rebuild your code. Then download and debug. Single-step through the code in the disassembly window until you reach the `STR` instruction with the comment "In memory view, goto 0x40025000 before execute." Before continuing in the debugger, go to the memory view, click inside the *Go To* field and enter `0x40025000`.

Now click step-into and execute the `STR R1, [R0]` instruction while observing the memory view window. *What happens?*

Continue single stepping through your code in the disassembly view until you return to the `CauseMemFault` section of code. Watch carefully as you single step through the code. *What happens?*

5 Lab Report

In the experiments sections above there are questions to be answered and directions to be followed in writing your lab report. Be sure that you have read through the experiments carefully, have taken notes, and have not missed anything.

Write a lab report that explains what you did and learned. Explain how the processor is designed to handle fault conditions. Explain the vector table. How does the stack work. What is context switching? What is the point of context switching and how does it work?

Finishing off with clock gating sets the stage for the next lab. You should now understand why the hard fault was caused in the first two experiments but not in the last. Explain this in your lab report.

And, as always, let's keep the focus on the subject of Computer Architecture. How does this lab relate to Computer Architecture?