EE 357 Lab 2–Array Processing

Prof. Annavaram

DUE: 5 PM., Friday, 10/12/2012

1 Introduction

In this lab you will write assembly code to store two different 4-bit even (binary) numbers from the DIP switches, and then echo the average of the two numbers. **Read carefully the procedure and requirements below.**

2 What you will learn

This lab is intended to teach you:

- How to iterate through arrays using postincrement(An)+ and predecrement–(An) modes.
- Implement appropriate control structures (loops and conditionals) with branching instructions to perform certain operations iteratively and to select appropriate data values
- To use the CodewarriorEclipse IDE to write assembly code and then use the integrated debugger to test your code. In addition, you will learn the basics of using your microcontroller's I/O pins as digital inputs and outputs (termed General Purpose I/O, or GPIO for short).

3 Background Information and Notes

Busy Looping: A very rudimentary method to have your processor wait for a specified amount of time is to enter a loop simply counting down from a specified start value until 0 is reached. Our MCF55259 processor runs at a frequency of 13.33 MHz. Assume that simple, 1-word arithmetic instructions [ADD/SUB Dn,Dm or ADDQ/SUBQ] take 1 clock cycle to execute while branches take 2. Moves or other instruction execution times depend on how many memory accesses are performed. Armed with this knowledge you should be able to write a loop that will take a specific amount of time to execute. Use this whenever we instruct you to wait for a specified period of time.

GPIO: Refer to the general purpose I/O notes from the previous lab.

Assembly-Only Projects: In this lab we will use an assembly-only project (i.e. no C code…using a 'main.s' rather than 'main.c'). Refer to the Eclipse setup document for notes on creating an assembly project.

4 Procedure

1. Create a new project (call it "ee357_lab2_io_arrays") and then import/setup the necessary assembly files into your project.
2. Open 'main.s' and use the following as a template.

```
            .data
NSAMP       .equ  3
arr1:       .space NSAMP
            .text
            .global _main
            .global main
            .include "../Project_Headers/ee357_asm_lib_hdr.s"
_main:
main:
            // Your code here
            // ...

            rts   // Last Line of Code
```

3. Your code must meet the following requirements:
   a. Take a 4-bit number from the 4-bit DIP switches; store the number in arr1[0]; and display it on the LED's until another number is set to the DIP switches – the second number must be different from the first number.
   b. Take the second 4-bit number and stores it in arr1[1].
   c. Display the second number about 10 seconds.
   d. Calculate the average of the two numbers and store it in arr1[2].
   e. Display the average number on the LED's for 1 second, then turn the LED's off for 1 second, and repeat forever (i.e. flash the average number on the LED's repeatedly)
4. Implementation requirements:
   a. You must setup a pointer to 'arr1' in address register A1.
   b. You must use postincrement and predecrement modes to iterate/reverse iterate through the arrays.
5. Build and then debug the project until it works as indicated above.
6. Demo your code to the TA.

5 Submission Instructions

(1) Turn in your working program (main.s) and review questions below (in a .pdf) on Blackboard **individually**.

(2) A date will be announced separately for demonstration.

6 Review questions

1. If we did not want to use postincrement and predecrement modes to traverse the arrays we could use our loop counters as an 'index' into the array via **A.R.I. w/ Index Register and Displacement Mode**.
   a. Create another project in a similar fashion as this one
   b. Copy your 'main.s' code
   c. Modify main.s to use A.R.I. w/ Index Register and Displacement Mode in place of any post-increment or pre-decrement modes.
   d. Ensure your new code works equivalently to your previous code
   e. Turn in a copy of this modified 'main.s' program on Blackboard.

Student Name: _____

Item	Outcome	Score	Max.
Demo Requirements: • Requirement 3. a • Requirement 3. b • Requirement 3. c • Requirement 3. d • Requirement 3. e	 Yes / No Yes / No Yes / No Yes / No Yes / No		 2 1 1 1 1
Implementation requirements (Code Examination): • You must setup a pointer to 'arr1' in address register A1. • You must use postincrement and predecrement modes to iterate/reverse iterate through the arrays.	 Yes / No Yes/No		 1 1
Index Mode Modification • (2 = Mostly correct,1 = Partially correct, 0 = Missing)			2
SubTotal			10
Late Deductions (30% per day)			
Total			
Open Ended Comments:			