**Lab 2: Using SPIM, Indexing and Looping**

for

David Retz

Computer Architecture

Course: CPE 315 Section: 01

California Polytechnic State University

San Luis Obispo, California

by

Patrick Cook

Brett Nelson

October 10, 2016

**Introduction:**

For lab 2 we were required to create two program in MIPS assembly that store and sum up tables of 32-bit and 8-bit signed integers.

**Purpose:**

The purpose of this lab is to become familiar with the MIPS simulator SPIM. More specifically, to learn how to access memory arrays, perform arithmetic functions, and do indexing in MIPS assembly language.

**Functional Requirements:**

Our assembly program must be able to:

* Store a table of 32-bit signed integers in memory
* Add the contents of above table to get a signed sum
* Print out the calculated sum to the console
* Store a table of 8-bit (byte) signed integers
* Similar to process with 32-bit integers, to sum up these integers and print out result

**Approach:**

In order to accomplish the goals of this lab we wrote two MIPS assembly programs, each corresponding to the type of integer being summed (32-bit or 8-bit).

To accomplish part 1 - adding 32-bit integers and printing out the sum - we wrote a program that would store the table of integers into words (32-bit data type). The program tracked the following variables to accomplish the goals mentioned above:

LENGTH - counter representing the number of integers, in words, to store and add

ARRAY - location of start of the sequence of integers, stored as words, to be added

str - part of the final string attached to the sum to be printed

SUM - the final value of the sum

A loop was then executed that fetched the next integer in the ARRAY (lw instruction), added that to the running SUM value (add instruction), incremented the address of next value to be fetched (addi instruction), decremented the LENGTH counter (by a word length via addi instruction), and finally branched back to the start of the loop if the counter was greater than zero (bne instruction).

Once the counter reaches zero, all values have been added. The SUM is then stored, and three syscall functions are executed to print out the str string, print out the sum, and end the program.

For part 2 - adding 8-bit integers and printing out the sum - the only difference is that byte data types are stored in data and added to the sum. As a result, ARRAY and LENGTH were stored in bytes, a lb (load byte) instruction was used to load each integer to be added, and the address was incremented by 1 byte, rather than 4.

**Difficulties during implementation:**

Our biggest difficulty in programming this lab was getting a feel for the SPIM simulation software. Learning the process for running assembly and encountering bugs through the SPIM application were tricky at first.

**Information learned from this lab:**

After completing this lab, we now have a strong understanding of the SPIM simulation environment and the tools it offers to program in assembly. We also got a nice overview of the various MIPS instructions and the format for programming in MIPS.

**Source Code:**

**lab2\_part1.s**

# Brett Nelson

# Lab 2: Using SPIM, Indexing and Looping (part 1)

# 10/8/16

#DEFINING GLOBAL DATA

.data # global data section

LENGTH: .word 5 # loop count

ARRAY: .word 4155543, 3112 , -2, 1054, -33543, 1233, -433433, 10101, 16384

str: .asciiz "Array Sum = " # string to print sum

SUM: .word 0 # location of the final sum

#DEFINING MAIN FUNCTION

.text # sets program location

.globl main # defines main program

main: lw $s0, LENGTH # loads loop length into $s0

la $t0, ARRAY # load address of ARRAY into $t0

and $s1, $s1, $zero # clear $s1

#LOOP

loop: lw $t1, 0($t0) # load the next value of array

add $s1, $s1, $t1 # add it to the running sum

addi $t0, $t0, 4 # increment address

addi $s0, $s0, -1 # decrement counter

bne $0, $s0, loop # loop back until complete

#STORE TOTAL

sw $s1, SUM # store the final total

#PRINTING THE SUM

la $a0, str # load string into a0 to be printed

li $v0, 4 # set syscall to print command

syscall

lw $a0, SUM # load sum into a0 to be printed

li $v0, 1 # set syscall to print command

syscall

li $v0, 10 # syscall to end program

syscall # ends program

.end

**lab2\_part2.s**

# Brett Nelson

# Lab 2: Using SPIM, Indexing and Looping (part 2)

# 10/8/16

#DEFINING GLOBAL DATA

.data # global data section

LENGTH: .byte 11 # loop count

ARRAY: .byte 40, 33, -127, 122, 4, 0, 16, 24, 32, -5, 123

SUM: .word 0 # location of the final sum

str: .asciiz "Array Sum = " # string to print sum

#DEFINING MAIN FUNCTION

.text # sets program location

.globl main # defines main program

main: lb $s0, LENGTH # loads loop length into $s0

la $t0, ARRAY # load address of ARRAY into $t0

and $s1, $s1, $zero # clear $s1

#LOOP

loop: lb $t1, 0($t0) # load the next value of array

add $s1, $s1, $t1 # add it to the running sum

addi $t0, $t0, 1 # increment address by only 1 byte

addi $s0, $s0, -1 # decrement counter

bne $0, $s0, loop # loop back until complete

#STORE TOTAL

sw $s1, SUM # store the final total

#PRINTING THE SUM

la $a0, str # load string into a0 to be printed

li $v0, 4 # set syscall to print command

syscall

lw $a0, SUM # load sum into a0 to be printed

li $v0, 1 # set syscall to print command

syscall

li $v0, 10 # syscall end program

syscall # ends program

.end