Lab 3 Report

Amber Raymer

Amberraymer0562@floridapoly.edu

EEL4768C: Computer Architecture and Organization

Due Date: 6 February, 2019

1.0 Project Description

Part a:

The purpose of this lab is to calculate a specific algebraic expansion without using any intrinsic multiplication instructions. The program accepts 3 positive integers a, b, and c as decimals and calculate f=[a+(a*b)+(a*b*c)] using a self-written multiplication routine. Lastly, the program will output f in decimal and binary.

Part b:

The purpose of this lab is to calculate a specific algebraic expansion without using any intrinsic multiplication or division instructions. The program accepts 3 positive integers a,b, and c as decimals and calculate g=[a+(a*b)+(a*b*c)] / max(a,b,c) using a self-written multiplicatoin and division routine. Lastly, the program will output g as a quotient with a remainder. We must indicate the denominator for the remainder.

Program Design

In the .data section, we have four asciiz values: enterText (our prompt), equation, decimal, and binary. In .text, we have many labels. Main asks for user input for a, b, and c. We also set two of our counters and then jump to firstMult where we use addition loops instead of multiplication for (a*b). We add "a+a" b times until the counter is greater than b. Once that condition is met, we jump to secondMult to do (a*b*c). In this section, I reuse the answer from firstMult and then just add (a*b) c times. If the counter equals c, we jump to calc. In this label, we add the three sections of our equation: a, (a*b), and (a*b*c) and then jump to print. Print displays the asciiz strings for the equation, decimal and binary and the corresponding answers are printed under them. Lastly, we terminate the program.

In part b, I was able to complete every part except for the division label. The only differences between part a and part b are the division, calcRemainder, print, maximum, t0Bigger, and t1Bigger labels. After the [a+(a*b)+(a*b*c)] is calculated from part a, we jump to maximum which uses bgt instructions to find the larger of two registers. Once we compare a, b, and c, we find the maximum and jump to division. I came very close to completing the division label. I believe I was on the right track with the logic. What I was attempting to do was add the maximum value to itself until it was larger than the numerator. After that, calcRemainder would subtract the maximum one time which would give us our quotient and our remainder. Lastly, we would print the answers.

Table 0a: MIPS Assembly Code (Part a)

lab1a.asm

```
# Amber Raymer -- 2/5/19
# lab3a.asm -- The purpose of this lab is to calculate a specific algebraic expansion without using any intrinsic multiplication instructions.
#		The program accepts 3 positive integers a,b, and c as decimals and calculate f=[a+(a*b)+(a*b*c)] using a self-written multiplicatoin routine.
#		Lastly, the program will output f in decimal and binary
# Registers used - $t0: First user input (a)
#		- $t1: Second user input (b)
#		- $t2: Third user input (c)
#		- $t3: f=[a+(a*b)+(a*b*c)]
#		- $t4: (a*b)
#		- $t5: (a*b*c)
#		- $t6: counter for "firstMult" loop
#		- $t7: counter for "secondMult" loop
#		- $v0: Syscall to read initial user input, return value
#		- $a0: Syscall parameter
# Resources: http://courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html (Syscall function parameters)
#		http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html (MIPS instruction set references)

.data
	enterText: .asciiz "Enter 3 positive integers for a, b, and c:\n"
	equation: .asciiz "f=[a+(a*b)+(a*b*c)] = \n"
	decimal: .asciiz "Decimal: "
	binary: .asciiz "\nBinary: "

.text
main:
	# Display the prompt to enter a, b, and c
	li $v0, 4
	la $a0, enterText
	syscall

	# Get first number from the user, put into $t0
	li $v0, 5
	syscall
	add $t0, $v0, $0	# Place a into $t0

	# Get second number from the user, put into $t1
	li $v0, 5
	syscall
	add $t1, $v0, $0	# Place b into $t1
```

```
        # Get third number from the user, put into $t2
        li $v0, 5
        syscall
        add $t2, $v0, $0        # Place c into $t2

        add $t6, $0, $0         # Set the first counter equal to 0
        add $t7, $0, $0         # Set the second counter equal to 0

        j firstMult


# This label uses addition loops instead of multiplication
firstMult:
        # $t4=(a*b) -> $t4 = $t0*$t1
        # $t6: counter for "secondMult" loop
        # Add "a + a" b times
        add $t4,$t4,$t0                 # Add a+a
        add $t6,$t6,1           # counter+=1
        beq $t6, $t1, secondMult        # If the counter is greater than b, go to second mult

        j firstMult                     # Else, continue loop

secondMult:
        # $t5=(a*b*c) -> $t5=$t4*$t2
        # $t7: counter for "secondMult" loop
        # Add "$t4 + $t4" $t2 times

        add $t5, $t5, $t4       # Add (a*b) c times
        add $t7, $t7, 1         # counter += 1
        beq $t7, $t2, calc      # Branch to calc if the counter equals $t4
        j secondMult            # Else, continue loop

calc:
        # Calculate f=[a+(a*b)+(a*b*c)]
        # $t3 = $t0 + $t4 + $t5
        add $t3, $t0, $t4       # $t3 = a + (a*b)
        add $t3, $t3, $t5       # $t3 = a+(a*b)+(a*b*c)
        j print

print:
        # Display the equation
        li $v0, 4
        la $a0, equation
        syscall
```

```
        # Display in decimal
        li $v0, 4
        la $a0, decimal
        syscall

        add $a0, $t3, $0        # f=[a+(a*b)+(a*b*c)]
        li $v0, 1               # Print $t3
        syscall

        # Display in binary
        li $v0, 4
        la $a0, binary
        syscall

        add $a0, $t3, $0        # f=[a+(a*b)+(a*b*c)]
        li $v0, 35              # Print $t3
        syscall



        j exit

exit:
        li $v0, 10              # Exit
        syscall
```

Table 0a: MIPS Assembly Code (Part b)

lab1b.asm

```
# Amber Raymer -- 2/5/19
# lab3b.asm -- The purpose of this lab is to calculate a specific algebraic expansion without using any intrinsic multiplication or division instructions.
#               The program accepts 3 positive integers a,b, and c as decimals and calculate g=[a+(a*b)+(a*b*c)] / max(a,b,c) using a self-written multiplicatoin and division routine.
#               Lastly, the program will output g as a quotient with a remainder. We must indicate the denominator for the remainder.
# Registers used - $t0: First user input (a)
#                - $t1: Second user input (b)
#                - $t2: Third user input (c)
#                - $t3: f=[a+(a*b)+(a*b*c)]
#                - $t4: (a*b)
#                - $t5: (a*b*c)
#                - $t6: counter for "firstMult" loop
#                - $t7: counter for "secondMult" loop
#                - $s0: counter for "div" loop
#                - $s1: max(a,b,c)
```

```
#                   - $s2: remainder
#                   - $s3: g=[a+(a*b)+(a*b*c)] / max(a,b,c)
#                   - $s4: max(a,b) (partial max)
#                   - $s5: quotient
#                   - $v0: Syscall to read initial user input, return value
#                   - $a0: Syscall parameter
# Resources: http://courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html (Syscall
function parameters)
#             http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html (MIPS instruction set
references)

.data
        enterText: .asciiz "Enter 3 positive integers for a, b, and c:\n"
        equation1: .asciiz "\nf=[a+(a*b)+(a*b*c)] = \n"
        equation2: .asciiz "\n\ng=[a+(a*b)+(a*b*c)] / max(a,b,c) = "
        quotient: .asciiz "\ng_quotient: "
        remainder: .asciiz "\ng_remainder: "
        decimal: .asciiz "Decimal: "
        binary: .asciiz "\nBinary: "

.text
main:
        # Display the prompt to enter a, b, and c
        li $v0, 4
        la $a0, enterText
        syscall

        # Get first number from the user, put into $t0
        li $v0, 5
        syscall
        add $t0, $v0, $0        # Place a into $t0

        # Get second number from the user, put into $t1
        li $v0, 5
        syscall
        add $t1, $v0, $0        # Place b into $t1

        # Get third number from the user, put into $t2
        li $v0, 5
        syscall
        add $t2, $v0, $0        # Place c into $t2

        add $t6, $0, $0         # Set the first counter equal to 0
        add $t7, $0, $0         # Set the second counter equal to 0

        j firstMult
```

```
# This label uses addition loops instead of multiplication
firstMult:
        # $t4=(a*b) -> $t4 = $t0*$t1
        # $t6: counter for "secondMult" loop
        # Add "a + a" b times
        add $t4,$t4,$t0                 # Add a+a
        add $t6,$t6,1           # counter+=1
        beq $t6, $t1, secondMult        # If the counter is greater than b, go to second mult
        j firstMult                     # Else, continue loop

secondMult:
        # $t5=(a*b*c) -> $t5=$t4*$t2
        # $t7: counter for "secondMult" loop
        # Add "$t4 + $t4" $t2 times
        add $t5, $t5, $t4       # Add (a*b) c times
        add $t7, $t7, 1         # counter += 1
        beq $t7, $t2, calc      # Branch to calc if the counter equals $t4
        j secondMult            # Else, continue loop

calc:
        # Calculate f=[a+(a*b)+(a*b*c)]
        # $t3 = $t0 + $t4 + $t5
        add $t3, $t0, $t4       # $t3 = a + (a*b)
        add $t3, $t3, $t5       # $t3 = a+(a*b)+(a*b*c)
        j maximum

maximum:
        # Find the maximum of a, b, and c
        bgt $t0, $t1, t0Bigger  # $t0 > $t1 -> t0Bigger
        add $s4, $t1, $0        # $t1 is bigger for now
        bgt $t2, $s4, t2Bigger # $t2 is biggest
        add $s1, $s4, $0        # max(a,b,c) = $t1
        add $s0,$s1,$0          # set counter
        j division

        t0Bigger:
                add $s4, $t0, $0
                bgt $t2, $s4, t2Bigger # $t2 is the maximum
                add $s1, $s4, $0        # max(a,b,c) = $t0
                add $s0,$s1,$0          # set counter
                j division

        t2Bigger:
                add $s1, $t2, $0        # max(a,b,c) = $t2
```

```
                add $s0,$s1,$0          # set counter
                j division

division:
        # $t3: f=[a+(a*b)+(a*b*c)]
        # $s3: g=[a+(a*b)+(a*b*c)] / max(a,b,c)
        # $s1: max(a,b,c)
        # $s5: quotient
        # $s2: remainder
        bgt $s1, $t3, calcRemainder   # Branch if the maximum value is greater than the
numerator
        add $s1, $s1, $s1             # max += max
        j division

        calcRemainder:
                sub $s1, $s1, $s1     # subtract to get the quotient
                add $s5, $s1, $0      # set quotient
                sub $s2, $t3, $s5     # numerator - quotient = remainder
                j print

print:
        # Display equation1
        li $v0, 4
        la $a0, equation1
        syscall

        # Display in decimal
        li $v0, 4
        la $a0, decimal
        syscall

        add $a0, $t3, $0        # f=[a+(a*b)+(a*b*c)]
        li $v0, 1               # Print $t3
        syscall

        # Display in binary
        li $v0, 4
        la $a0, binary
        syscall

        add $a0, $t3, $0        # f=[a+(a*b)+(a*b*c)]
        li $v0, 35              # Print $t3
        syscall

        # Display equation2
        li $v0, 4
```

```
        la $a0, equation2
        syscall

        # Display in decimal
        li $v0, 4
        la $a0, quotient
        syscall

        # $s5: quotient
        # $s2: remainder

        add $a0, $s2, $0        # f=[a+(a*b)+(a*b*c)] / max(a,b,c)
        li $v0, 1               # Print $s3
        syscall

        # Display in binary
        li $v0, 4
        la $a0, remainder
        syscall

        add $a0, $s2, $0        # remainder
        li $v0, 1               # Print $s2
        syscall


        j exit

exit:
        li $v0, 10              # Exit
        syscall
```

2.0 Symbol Table

The registers used in the MIPS code and their purpose are listed below. The labels used in the MIPS code are also identified and described. The data type is also identified where applicable. Table 1 is a symbol table for the registers used and Table 2 is a symbol table for the labels used.

Table 1a: Registers Used

Register	Use
$v0	Tells MARS what to do when a syscall is called and used to return values.
$a0	The argument to be used during some syscalls. Also used to pass arguments depending on the parameter used.
$t0	Holds the first integer from the user input at runtime.

$t1	Holds the second integer from the user input at runtime.
$t2	Holds the third integer from the user input at runtime.
$t3	Holds the answer to f=[a+(a*b)+(a*b*c)].
$t4	Holds the answer to (a*b).
$t5	Holds the answer to (a*b*c).
$t6	This is used as a counter for “firstMult”.
$t7	This is used as a counter for “secondMult”.

Table 1b: Registers Used

Register	Use
$v0	Tells MARS what to do when a syscall is called and used to return values.
$a0	The argument to be used during some syscalls. Also used to pass arguments depending on the parameter used.
$t0	Holds the first integer from the user input at runtime.
$t1	Holds the second integer from the user input at runtime.
$t2	Holds the third integer from the user input at runtime.
$t3	Holds the answer to f=[a+(a*b)+(a*b*c)].
$t4	Holds the answer to (a*b).
$t5	Holds the answer to (a*b*c).
$t6	This is used as a counter for “firstMult”.
$t7	This is used as a counter for “secondMult”.
$s0	This is a counter for “division”.
$s1	Max(a, b, c)
$s2	Remainder for “division”.
$s3	g=[a+(a*b)+(a*b*c)] / max(a,b,c)
$s4	Partial maximum (a, b)
$s5	Quotient for “division”.

Table 2a: Labels Used

Label	Use
main	Executed first at runtime.
firstMult	Executes (a*b).
secondMult	Executes (a*b*c).
calc	Calculates f=[a+(a*b)+(a*b*c)].
print	Displays the equation and the decimal and binary equivalent answer.
exit	Terminates the program.

Table 2b: Labels Used

Label	Use

main	Executed first at runtime.
firstMult	Executes (a*b).
secondMult	Executes (a*b*c).
calc	Calculates f=[a+(a*b)+(a*b*c)].
print	Displays the equation and the decimal and binary equivalent answer.
exit	Terminates the program.
maximum	Finds the maximum of a, b, and c.
t0Bigger	Sets $t0 as the maximum.
t1Bigger	Sets $t1 as the maximum.
division	Uses subtraction and addition to "divide" by the maximum.
calcRemainder	Sets the quotient and remainder.

3.0 Learning Coverage

The skills implemented during this lab include:

- Algebraic expansion without using intrinsic multiplication and division instructions
- Understand tradeoffs of HW vs. SW
- Printing decimal and binary numbers

4.0 Test Plan

To properly test this code, the user must input three positive integers. If all test cases give proper outputs, the program functions correctly.

Table 4a: Test Cases Used (Part a)

Test Case	Pixel Data	Expected Output
Standard/random	$t0 = 5 $t1 = 19 $t2 = 21	Decimal: 2095 Binary: 00000000000000000000100000101111
Maximum values	$t0 = 2,147,483,647 $t1 = 2,147,483,647 $t2 = 2,147,483,647	Runtime exception at 0x00400014: invalid integer input (syscall 5)
Minimum values	$t0 = 0 $t1 = 0 $t2 = 0	The program requires POSITIVE integers.
Most bits different	$t0 = 131,071 $t1 = 0 $t2 = 0	Runtime exception at 0x00400040: arithmetic overflow

Table 4b: Test Cases Used (Part b)

Test Case	Pixel Data	Expected Output
Standard/random	$t0 = 5 $t1 = 19 $t2 = 21	Decimal: 2095 Binary: 00000000000000000000100000101111
Maximum values	$t0 = 2,147,483,647 $t1 = 2,147,483,647 $t2 = 2,147,483,647	Runtime exception at 0x00400014: invalid integer input (syscall 5)
Minimum values	$t0 = 0 $t1 = 0 $t2 = 0	The program requires POSITIVE integers.
Most bits different	$t0 = 131,071 $t1 = 0 $t2 = 0	Runtime exception at 0x00400040: arithmetic overflow

5.0 Test Results

An image depicting three successful (a) and unsuccessful (b) test runs of the program is shown below.

Table 5a: Test Cases – Part a

Test Cases – Part a	
Standard/random	Enter 3 positive integers for a, b, and c: 5 19 21 f=[a+(a*b)+(a*b*c)] = Decimal: 2095 Binary: 00000000000000000000100000101111 -- program is finished running --

Table 5b: Test Cases – Part b

Test Cases – Part a

Standard/random	Enter 3 positive integers for a, b, and c: 5 19 21 f=[a+(a*b)+(a*b*c)] = Decimal: 2095 Binary: 00000000000000000000100000101111 g=[a+(a*b)+(a*b*c)] / max(a,b,c) = g_quotient: 2095 g_remainder: 2095 -- program is finished running --

6.0 References

6.1 Lecture Notes

EEL 4768C Modules from lectures: Slides adapted from Dr. DeMara (UCF), Dr. Sahawneh (FPU), Dr. Mutlu (CMU). They are available on Canvas.

6.2 MARS Simulator

The MARS Simulator for MIPS processors, available at:
http://courses.missouristate.edu/kenvollmar/mars/

6.3 MARS Syscall Reference Guide

http://courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html

7.4 MIPS Instruction Set Reference Sheet

http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html