Computer Architecture
&
Organization (EEL 4768)

Lab #3

Exercise[1]:

Part A

Overview

The primary objective of these exercises is to increase your understanding of the fundamental instructions that hardware uses to execute a user program. To illustrate HW vs. SW tradeoffs, a few arithmetic calculations shall be performed using loops, rather than using intrinsic hardware circuits for multiplication and division. So use of loops along with branch operations will be practiced. You will also learn MIPS hardware instructions sufficient to handle common ALU tasks.

Guideline: Write MIPS code for Triplex Algebraic Expansion

You are tasked to calculate a specific algebraic expansion, i.e. compute the value of **`f`** for the expression

`f=[a+(a*b)+(a*b*c)]`

without using any intrinsic multiplication instructions. More formally, write MIPS assembly code that accepts three positive integers **`a`**, **`b`**, and **`c`** as input parameters. The code shall execute in MARS to prompt the user to enter three positive integers represented in decimal, each separated by the **`Enter`** key. The program shall calculate **`f=[a+(a*b)+(a*b*c)]`** using your own self-written multiplication routine. The program will then output **`f`** in decimal and binary, using **`syscall`** routines for each output.

Note: To receive credit, no multiplication, no division, and no shift instructions shall be used. Namely, none of {**`mul, mul.d, mul.s, mulo, mulou, mult, multu, mulu, div, divu, rem, sll, sllv, sra, srav, srl, srlv`** } or else a zero score will result. Thus, it is necessary to compose your own multiplication technique. In addition, use of a loop is ***required*** for credit to realize the multiplication code. For uniformity of grading and to focus on practicing the current topics, any use of **`jal/jr`** instructions is ***disallowed***: a score of zero will result if **`jal/jr`** are used.

Hint: MARS supports output in binary format using the appropriate parameters with **`syscall`**. For the C-language prototype, there are many approaches to print a value in binary in C. Here is my favorite single-line-in-a-loop solution that you can use with any C-compiler to print **`x`** in binary, shown with **`x=9`** as an example:

```
main(){
int i,x=9;
for (i=31;i>=0;i--) {
      /* put char with ASCII code for either 0 or 1 by adding 1 for the ith bit to ASCII of '0' */
      putchar('0' + ((x>>i) & 1));
     }
return(0);
}
```

You can use the above in your C-code prototype or write your own binary output routine in C if you want.

[1] This project has been inspired and adopted from Dr. DeMara's projects at UCF

Part B

You are tasked to use the same positive integers from Part A to also compute:

`g = [a+(a*b)+(a*b*c)] / max(a,b,c)`

More formally, write MIPS code to output the result of above expression of **`g`** without using any built-in MIPS/MARS instructions for multiplication or division. The values already entered for Part A for **`a`**, **`b`**, and **`c`** shall be used. Output the value of **`g`** in {quotient with remainder} format as separate decimal integers. Indicate the denominator for the remainder.

Note: To receive credit, no multiplication, no division, and no shift instructions shall be used. Namely, none of {**`mul, mul.d, mul.s, mulo, mulou, mult, multu, mulu, div, divu, rem, sll, sllv, sra, srav, srl, srlv`**} or else a zero score will result. Thus, it is to compose your own division technique. In addition, use of a loop is ***required*** for credit to realize the division code. It part of the project points to design a way to realize division using a loop. For uniformity of grading and to focus on practicing the current topics, any use of **`jal/jr`** instructions is ***disallowed***: a score of zero will result if **`jal/jr`** are used.

Hint: You can refer to the definition of division and how division works. For example, given a positive integer *X*, and a positive integer *Y* where *X*>*Y* then the division *X*/*Y* is computed such that unique integers *Q* and *R* satisify $X=(Y * Q + R)$ *where* $0 \leq R < Y$. The value *Q* is called the quotient and *R* is called the remainder. Some examples are:

{X = 7, Y = 2} then 7 = 2 * 3 + 1 so Q=3 and R=1
{X = 8, Y = 4} then 8 = 4 * 2 + 0 so Q=2 and R=0
{X = 13, Y = 5} then 13 = 5 * 2 + 3 so Q=2 and R=3

Sample Program Output

Sample output for Part A is:

```
Enter 3 integers for a,b,c respectively:
9
15
21
f_ten = 2979
f_two = 00000000000000000000101110100011
```

Part B would additionally provide the following outputs:

```
g_quotient = 141
g_remainder = 18 / 21
```

Your program shall provide outputs in **exactly the sequence and format shown above**. To receive full credit, no additional superfluous outputs shall occur.