Bradley Grose

CSCE 212 Fall 2020

Dr. Wang

Project 1

1.0 Project Description

This report outlines the work I did for Project 1 for CSCE 212. The purpose of this code is to be able to use MIPS programming language to develop code that will ask the user to input 4 integer numbers separated by enter. Using the MIPS functionality, the code takes in those values and using the first 4 non-zero digits of our student ID, does calculations detailed later.

In my case, my USC Student ID is F14304441, meaning the first 4 non-zero numbers are 1, 4, 3, 4, which I will be referring to as A, B, C, D respectively. The code must first solve for f and g, the user will input 4 numbers: x, y, z, w respectively. Then, the following math must be done $f=(x^A+y^B)$ and $g=(z^C+w^D)$, which equivalates to $f=(x^1+y^4)$ and $g=(z^3+w^4)$. Then, that is printed out as well as its binary equivalent.

Then, for part 2, the code must divide f by g. It should give the quotient and the remainder as an answer. All math must be addition of subtracting.

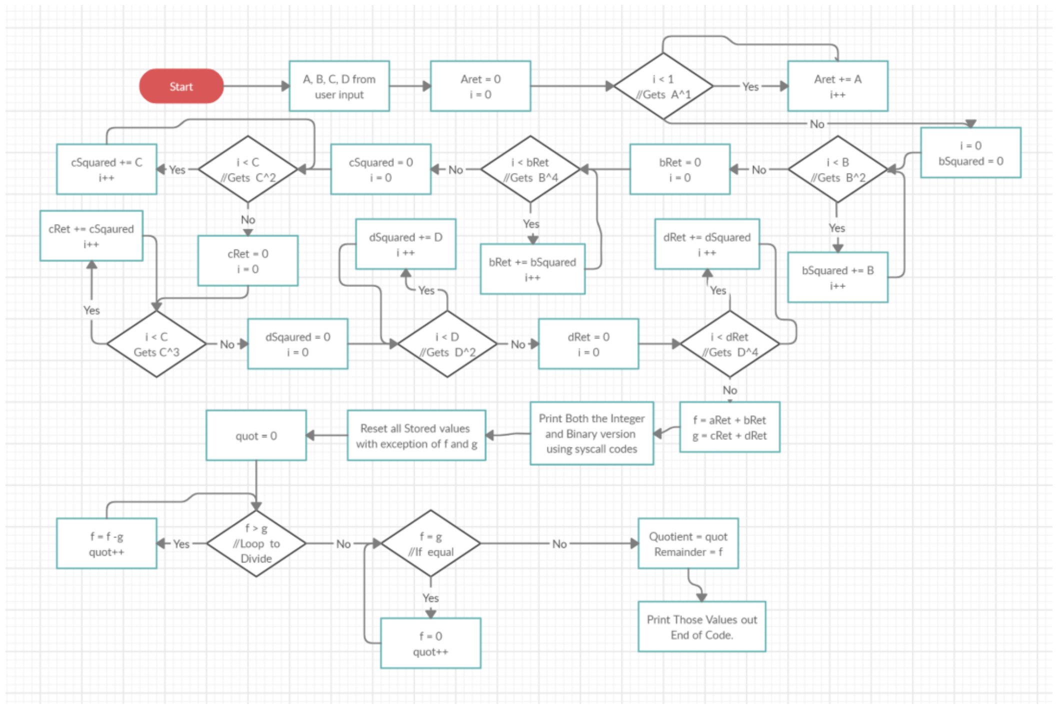
2.0 Program Design

Because the only math operation that can be used is addition and subtraction, I utilized loops to create a way of solving the problem. First, I print out my Student ID and instructions on inputting the 4 variables for the math. The user should input 4 integers, separated by an enter.

The first thing the code will do is create a temporary variable and goes through a loop to add that number once to it, as the first number is only raised to the first power. However, this loop code is the same as the other loops and I wanted to design the functionality. For B, I had to raise it to the 4th power. So what I did was make a squared temp variable to store the value. Then I run a for loop using I start at 0 and while it is less than the value of B, it will add B to the squared storing variable. Each time it runs through it will add 1 to I. Then, once the loop is done, I run the same style of loop, however this time I has to be less than the squared value holder. This allows me to step over the process of cubed and go straight to 4th power, as addition works that way for doubling the power. This runs the same loop; however it creates a return value for B and adds the squared value each time through the loop. This same process is used for D. For C however, because I want the cubed value of it, I will run the first step twice to get the cube value of adding C to the store value for 2*C times.

Then, once I have A^1 B^4 C^3 D^4, I add the value of A^1 and B^4 to get f. I then add C^3 and D^4 together to get g. Then, using syscall functions, I print out f and g, both as integer using syscall with code 1 and as binary using syscall with code 35. After that, I reset all the values except for where f and g are stored to 0 for part 2.

For part 2, I had to solve what f/g is equal to with quotient and remainder. For the quotient, I created a counter value for it starting at 0. I once again used a loop that checks to see if f is greater than g. If it is, it will enter the loop. In the loop, it will subtract g from f and store that as f (f=f-g). Then it will add one to the value of quotient. Once the loop is no longer true, I check to make sure that f and q are not equal. If they are, I add one to the quotient value and then set f to 0, as f will be used to print out the remainder. Then, using syscall again, I print out the string to label the values, one with the quotient value and one with the remainder value which is where f is stored at this point. This will then conclude the running of my code. The flowchart below shows the process of the code.



3.0 Symbol Table

Part 1 Symbols

Registers	Purpose/Where Its Used
string1	String value for that prints out my student ID. Used at beginning to print out command.
string2	String value that prompts the user to enter values for ABCD. Used at beginning to print command.
decimal1	String value to label the decimal value of f. Used when f is printed out.
binary1	String value to label the binary value of f. Used when f is printed out.
decimal2	String value to label the decimal value of g. Used when f is printed out.
binary2	String value to label the binary value of g. Used when f is printed out.
$v0	Used to store the syscall int value and intake syscall values. Used whenever I print out a string or value and take in 4 number
$a0	Sends the value in to be printed out. Used when I print instructions or the results
$t0	Used to store the original value of A inputted by the user.
$t1	Used to store the original value of B inputted by the user.
$t2	Used to store the original value of C inputted by the user.
$t3	Used to store the original value of D inputted by the user.
$t4	This is used as the I value for the for loops for raising the inputted variables to a power. This is used in all of the loops below.
$s0	Used for 2 different things. At first it is used to store the value of A^1 power in ALoop1, however later in the code it is used to store the value of f.
$s1	Used for 2 different things. At first it is used to store the value of B^2 power in BLoop1, however later in the code it is used to store the value of g.
$s2	Used to store the value of B^4 power in the Bloop2 label. This is also used when adding to get f.
$s3	Used to store the value of D^2 power in the Dloop1 label. This is also used when getting D^4.

\$s4	Used to store the value of D^4 in DLoop2. This is also used when adding for g.
\$s5	Used to hold the value of C^2 in the CLoop1 label area. This is also used in CLoop2 to calculate C^3.
\$s6	Used to store the value of C^3 in CLoop2 as well as adding to get the value of g.
ALoop1	Label used to loop to get the value of A^1 by running a for loop.
AQuit1	Label called as an exit to ALoop1 that resets the I value in \$t4.
BLoop1	Label used to loop to get the value of B^2 by running a for loop.
BQuit1	Label called as an exit to BLoop1 that resets the I value in \$t4.
BLoop2	Label used to loop to get the value of B^4 by running a for loop.
BQuit2	Label called as an exit to BLoop2 that resets the I value in \$t4.
CLoop1	Label used to loop to get the value of C^2 by running a for loop.
CQuit1	Label called as an exit to CLoop1 that resets the I value in \$t4.
CLoop2	Label used to loop to get the value of C^3 by running a for loop.
CQuit2	Label called as an exit to CLoop2 that resets the I value in \$t4.
DLoop1	Label used to loop to get the value of D^2 by running a for loop.
DQuit1	Label called as an exit to DLoop1 that resets the I value in \$t4.
DLoop2	Label used to loop to get the value of D^4 by running a for loop.
DQuit2	Label called as an exit to DLoop2 that resets the I value in \$t4.

All Values with the exception of \$s0 and \$s1 are reset to 0 going into part 2. This is because I still need the value of f and g for part 2

Part 2 Symbols

Registers	Purpose/Where Its Used
$s0	Used to store the value of f. This changed in the LoopQuot where it is subtracted from to find the quotient. This is later printed out as the remainder
$s1	Used to store the value of g. This is never changed however it is used to subtract from f.
$t0	Original set at 0, this stores the value that will be printed out as the quotient. This gets one added to it every time LoopQuot runs.
$v0	Used to store the syscall int value. Used whenever I print out a string or value.
$a0	Sends the value in to be printed out. Used when I print the results.
quotient1	The String to print out before the quotient. Used when printing at end.
remainder1	The String to print out before the remainder. Used when printing at end.
LoopQuot	Loop that runs to subtract the value of g from f as long as f is greater than g
Done	Exits out of the loop, to then check one condition
Same	Called if g is equal to f. This adds one to quotient and sets the remainder to 0
DoneFin	Called when all math is done. Leads to printing of results from the math done.

4.0 Learning Coverage

Some technical topics learned and used in this project are below

1. Using only addition in order to raise a number to a power and making it not purely hard coded
2. Loop Control in MIPS using BEG to make FOR loops
3. Using Loops to divide numbers while only using subtraction to do the math
4. Using li to output a stored integer as a binary number using code 35
5. How to make an if statement using MIPS code using beq

5.0 Test Plan

For my testing, I plan on running three different tests in order to test the functionality of the code. I will hand solve/use a calculator to evaluate the expected result and compare to the result. I will make sure to test multiple scenarios, such as large numbers, 0, as well as ones where the expected quotient or remainder should be 0. This will cover all different areas of the code to assure that the math is being done correctly and every label will be tested using these 3 processes. I will hand check each binary value Below are the 3 plans:

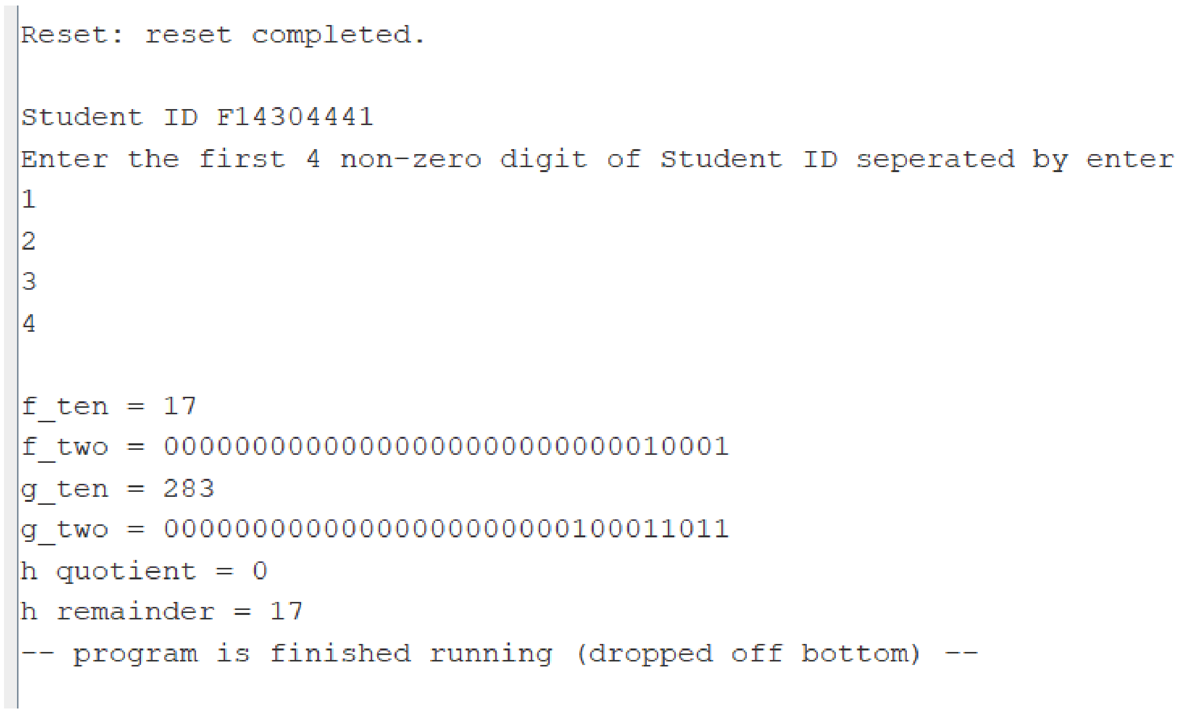
Plan 1: Send in 1, 2, 3, 4. F expected 17, G expected is 283, Quotient is 0, remainder is 17.

Plan 2: Send in 5, 5, 3, 0. F expected 630, G expected is 27, Quotient is 23, remainder is 9.

Plan 3: Send in 73, 2, 2, 3. F expected 89, G expected is 89, Quotient is 1, remainder is 0.

6.0 Test Results

Test1: Successful

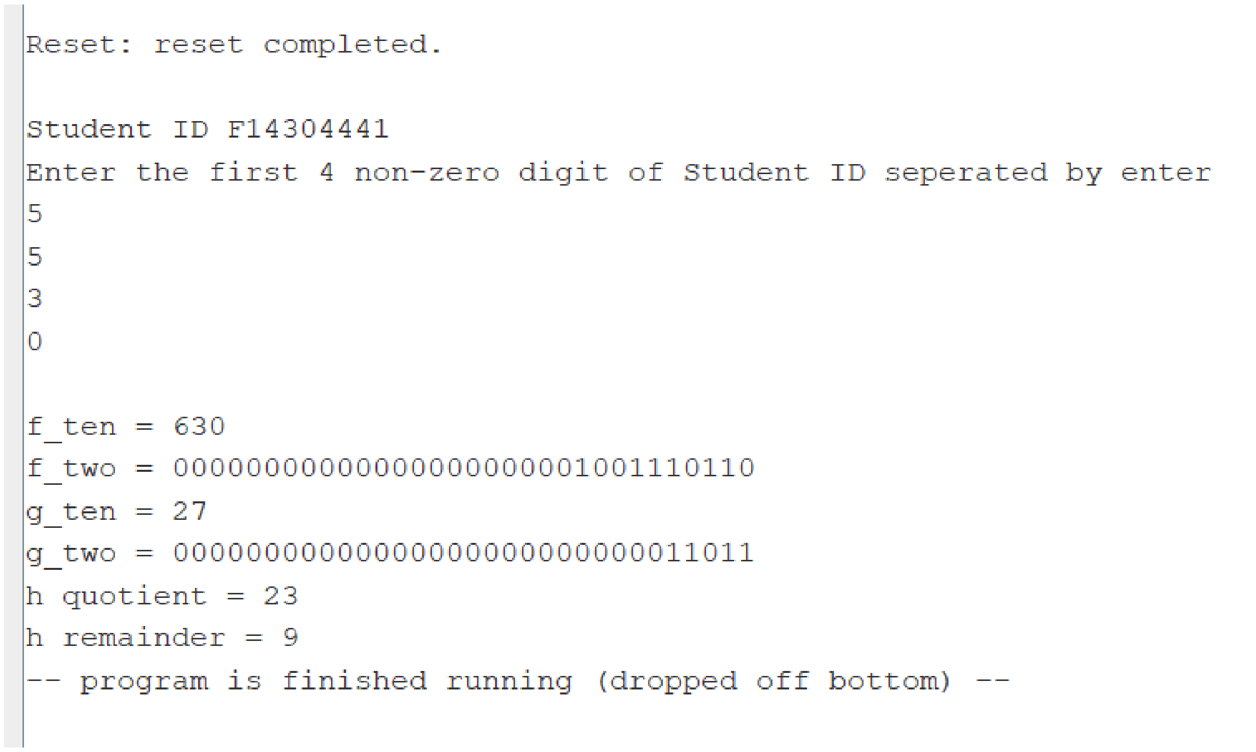


```
Reset: reset completed.

Student ID F14304441
Enter the first 4 non-zero digit of Student ID seperated by enter
1
2
3
4

f_ten = 17
f_two = 00000000000000000000000000010001
g_ten = 283
g_two = 00000000000000000000000100011011
h quotient = 0
h remainder = 17
-- program is finished running (dropped off bottom) --
```

Test2: Successful

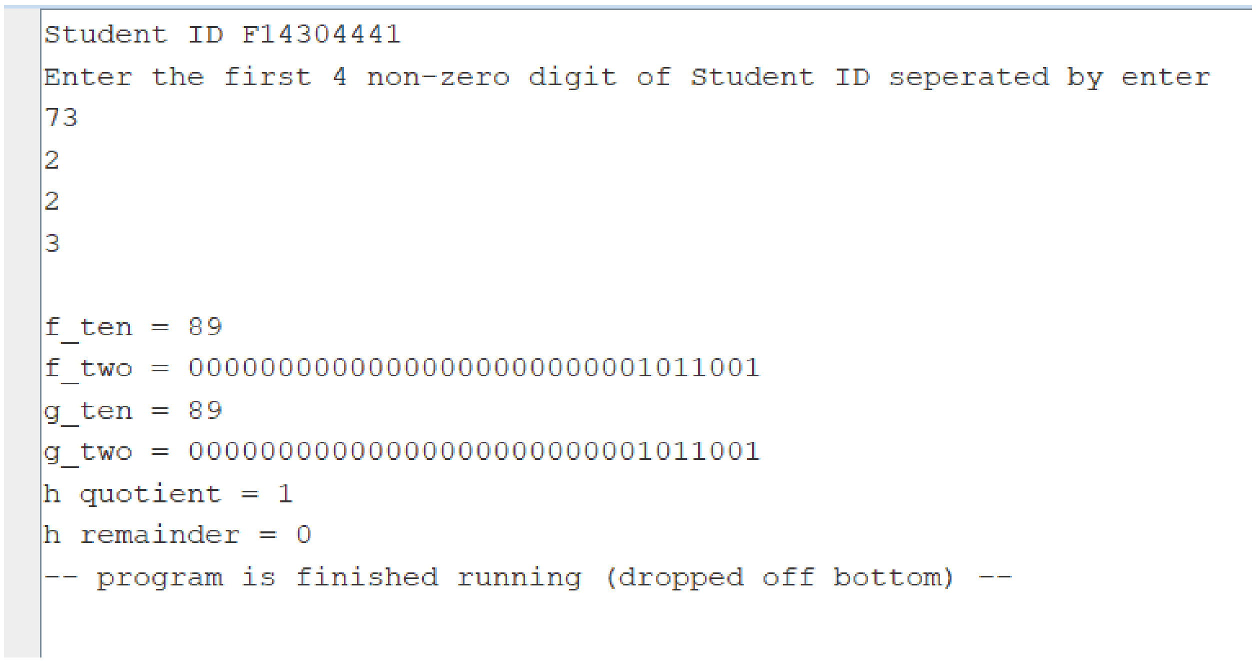


```
Reset: reset completed.

Student ID F14304441
Enter the first 4 non-zero digit of Student ID seperated by enter
5
5
3
0

f_ten = 630
f_two = 00000000000000000000001001110110
g_ten = 27
g_two = 00000000000000000000000000011011
h quotient = 23
h remainder = 9
-- program is finished running (dropped off bottom) --
```

Test3: Successful



```
Student ID F14304441
Enter the first 4 non-zero digit of Student ID seperated by enter
73
2
2
3

f_ten = 89
f_two = 00000000000000000000000001011001
g_ten = 89
g_two = 00000000000000000000000001011001
h quotient = 1
h remainder = 0
-- program is finished running (dropped off bottom) --
```