Mitchell Jonker
CSCE-212-001
1 March, 2022

Project 2 Report

1.0 | Program Inputs and Outputs:

1.1 | Program 1 uses system output to the command terminal, and takes in information through the command terminal as well.

1.2 | Program 2 also uses system output to the command terminal and takes user input from the command terminal. This program also uses floating point procedures and registers to operate with single precision floating point data operations.

1.3 | Program 3 uses command line input and output, as well non-leaf procedural operations, utilizing jump-and-link functionality, where the stack is used to preserve the return address.

2.0 | Program Design:

2.1 | Program 1 first uses system call operations to prompt the user to enter three values, as well as store those values in registers. The goal of this program is to determine which value is the smallest non-zero of the three entered values.Then, using a series of branching (and leaf) operations are applied. This logic system rearranges the stored registers of the inputted values until they are sorted from smallest to largest. This system is functionally designed around accepting three values and sorting them. After being sorted, the program checks to make sure the smallest value is not zero. If this smallest value is not equal to zero, the program prints the minimum value and terminates. However, if the smallest value is zero, the three values are shifted, and the second smallest value is tested. If it's also a zero, the program then attempts to try the third. If at any point the succeeding value is not equal to zero, it is printed and the user is notified that it is the smallest value, since it is the smallest non-zero value. If all three values are 0, the program notifies the user and terminates.

2.2 | Program 2 intakes a person's height and weight, then determines the individual's BMI, and notices if the BMI indicates that the individual is underweight, normal weight, or overweight. This program uses single precision floating point data operations for an additional layer of accuracy . This means that different instructions will be used than when integers are used. Preset float variables are defined to determine if the BMI indicates underweight, normal weight, or over weight. The user is prompted to enter the individual's weight in pounds, and height in feet. These values are stored in initialized float variables. Feet are then converted to inches, and then squared. The BMI calculation is then executed. The weight is divided by the inches squared, and then that resultant value is multiplied by 703. This resultant value is the BMI of the individual. This value is contrasted with the bounds for underweight and overweight. If the value of the BMI is less than 18.5, the individual is underweight, and the user is notified of the BMI value and the weight status. If the BMI value is greater than or equal to 25, the individual is overweight,

and the user is notified. Otherwise, the user has a normal weight, and the user is notified. After being notified of whichever result, the program resets, prompting the user to enter new values for another individual.

2.3 | Program 3 serves to calculate how much time overall time it takes for an individual to complete their homework assignments and exercises. The program prompts the user to enter (and then stores) 1) How many homework the individual completed, 2) on average how long it took to complete one homework, 3) how many exercises the individual completed, 4) and on average how long it took to complete one exercise. These values are stored in temporary registers. Then a jump-and-loop process is initiated where the overall time for homeworks, and overall time for exercises are calculated with the assistance of the stack and non-leaf procedure. After this, the total time is stored in a saved register, and then the user is notified of the total time.

3.0 | Symbol Table:

Register	Purpose & Labels (Program 1)	Purpose & Labels (Program 2)	Purpose & Labels (Program 3)
$a0	Syscall address holder. Holds the address of what register will be used in syscall function	Syscall address holder. Holds the address of what register will be used in syscall function	Syscall address holder. Holds the address of what register will be used in syscall function
$v0	Syscall behavior value holder. Syscall function acts differently based on what value is held here.	Syscall behavior value holder. Syscall function acts differently based on what value is held here.	Syscall behavior value holder. Syscall function acts differently based on what value is held here.
$s0	Stores first entered value, eventually holds lowest value after being sorted	-	Fetches total time from the stack and saves it.
$s1	Stores second entered value, eventually holds second lower value after being sorted	-	-
$s2	Stores third entered value, eventually holds largest value after being sorted	-	-
$t0	Used to store values temporarily through sorting process.	-	Holds integer: number of homeworks completed
$t1	-	-	Holds integer: average time per homework

$t2	Loop counter; value increases per check - insures that if every value entered is 0, the user is notified that no non-zero value was entered.	-	Holds integer: number of exercises completed
$t3	-	-	Holds integer: average time per exercise
$t4	-	-	Value holder used when interacting with the stack.
$t5	-	-	Value holder used when interacting with the stack.
$t6	-	-	Stores total time into stack
$f0	-	Holds single precision value: float variable read from syscall function	-
$f1	-	Holds single precision value: weight of individual	-
$f2	-	Holds single precision value: height of individual, converted to inches from feet	-
$f3	-	Holds single precision value: pounds per inches squared (of individual's weight & height)	-
$f4	-	Holds temporary constant to perform calculations like: feet to inches, and applying the BMI metric constant	-
$sp	-	-	Stack pointer register. Used to adjust where in the stack we are wishing to operate with.
$ra	-	-	Return address register. When jump-and-linking, this register stores the address of the next

4.0 | Learning Coverage:

4.1 | I learned how to use procedures to more easily program in MIPS.

4.2 | I learned how to use the jal function and how to works with the $ra register to code in a non-leaf format.

4.3 | I learned how to interact with the stack, storing and reading values in the stack, and operating with the $sp register to adjust the stack pointer, a crucial element in furthering the use of MIPS.

4.4 | I learned how to use float registers, specifically with single precision floating point variables. This includes the dedicated commands and coprocessor for SPFPV's.

4.5 | I learned more syscall functions, allowing further functionality within MIPS. Specifically using new syscall values to read and print floats.

5.0 | Test Results

5.1 | Program 1 Test Results:

```
Welcome to Program 1!
Enter three numerical values, each separated by an Enter key.
6
0
3
The lowest non-zero number is: 3
-- program is finished running --
```

```
Welcome to Program 1!
Enter three numerical values, each separated by an Enter key.
0
3
8
The lowest non-zero number is: 3
-- program is finished running --
```

5.2 | Program 2 Test Results:

```
Welcome to the BMI calculator!
Enter the weight (in pounds).
205
Enter the height (in feet).
5.9
BMI #: 28.750317
BMI Index is Overweight.


Enter the weight (in pounds).
170
Enter the height (in feet).
6
BMI #: 23.053625
BMI Index is Normal.


Enter the weight (in pounds).
```

```
Welcome to the BMI calculator!
Enter the weight (in pounds).
150
Enter the height (in feet).
5.3
BMI #: 26.069477
BMI Index is Overweight.


Enter the weight (in pounds).
225
Enter the height (in feet).
7
BMI #: 22.417091
BMI Index is Normal.


Enter the weight (in pounds).
```

5.3 | Program 3 Test Results:

```
Welcome to Total Work Time Calculator!
How many homeworks did you complete?
5
On average, how long did it take to complete one homework (in hours)?
2
How many exercises did you complete?
3
On average, how long did it take you to complete one exercise (in hours)?
4
The total work time in hours: 22
-- program is finished running (dropped off bottom) --
```

```
Welcome to Total Work Time Calculator!
How many homeworks did you complete?
2
On average, how long did it take to complete one homework (in hours)?
1
How many exercises did you complete?
4
On average, how long did it take you to complete one exercise (in hours)?
2
The total work time in hours: 10
-- program is finished running (dropped off bottom) --
```