Mitchell Jonker
CSCE-212-001
4 April, 2022

Project 3 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 accepts user input through the command terminal.

1.3 | Program 3 utilizes floating point procedures and registers to operate with single precision floating point data operations. It also uses system input and output through the command terminal.

2.0 | Program Design:

2.1 | Program 1 first stores the values for A and B, then identifies which is larger. Since multiplication procedures are not used in this process, an addition loop is used to add A to itself B-many times. This uses a temporary value holder of what A originally was, and a temporary incremental counter. This loop continues until the incrementing counter is equal to B. At this point the process terminates, and the user is returned the value of A*B (by adding A to A B-many times. There are some weird conditions that take place when using this sort of loop, so there are exceptions in place for conditions if either of the values equal 0, 1, or 2.

2.2 | Program 2 stores A, B, and C in temporary registers. Then, each part of the function f is calculated. First, the B^C is calculated. This is done by multiplying B*B C-many times. After this, A is added to B^C. This is the first third of the equation. Next a temporary value for a^2 is calculated, and c has that value subtracted from it. This is the second third. This c-a^2 is then checked. If it is equal to zero, then the program will throw a divide by zero exception and end any further calculations. As long as that expression is not equal to zero, the program continues. The final third of the function F is calculated by multiplying by the immediate value 3. Now that each individual element of F has been calculated, each is tested for overflow. This is performed by checking the to see signs of the two elements, and then the resulting sign after the operation performed on the values. If they are logically correct, then no overflow is present. This is tested using a overflow procedure, that is passed in the same temporary values, but before testing each function, each function's components and their result are passed into the checking registers. If no overflow exceptions are thrown, the values are combined into the function F and printed (with a notice of if any overflow was noticed), then the program exits.

2.3 | Program 3 saves the IC, CPI, and Clock Rate for processors A and B. These values are stored as floating point single precision variables. These FP values are then calculated into the CPU Time for each processor, and then compared. To check if one is greater than the other, they are first translated into temporary integer values, and then passed into the register from the coprocessor. If the time is greater for processor A, processor B is faster, so processor A's cpu time is divided by the processor B's time, and that is how much faster processor B is than processor A. The same process occurs for when processor B's time is greater than processor A's time, only each swaps spots in the relative performance speed equation. Additionally, if the CPU times for both processors are equal, the user is notified of such. Finally, the program quits after notifying the user of the performance difference (or equality) of the processors.

3.0 | Symbol Table:

Register	Purpose & Labels (Program 1)	Purpose & Labels (Program 2)	Purpose & Labels (Program 3)
$a0	Holds the address or label of data for syscall functions	Holds the address or label of data for syscall functions	Holds the address or label of data for syscall functions
$v0	Integer held here determines the behavior of syscall function	Integer held here determines the behavior of syscall function	Integer held here determines the behavior of syscall function
$t0	Inputted Variable A	Inputted Variable A	-
$t1	Inputted Variable B	Inputted Variable B	-
$t2	Loop Counter	Inputted Variable C	-
$t3	Temporary sorting hold	Temporary counter	-
$t4	Reference of largest value	Hold original B for loop	-
$t5	-	Overflow Test Element 1	-
$t6	-	Overflow Test Element 2	-
$t7	-	Overflow Logical Register	-
$t8	-	Overflow Test Element 3	-
$t9	-	Temporary counter hold	-
$f0	Conversion of answer to FP	-	-
$f1	-	-	Processor A: IC
$f2	-	-	Processor A: CPI
$f3	-	-	Processor A: CR
$f4	-	-	Processor B: IC
$f5	-	-	Processor B: CPI

$f6	-	-	Processor B: CR
$f7	-	-	Processor A: CPU Time
$f8	-	-	Processor B: CPU Time
$f9	-	-	Temporary integer version of $f7
$f10	-	-	Temporary integer version of $f8
$f11	-	-	Relative Performance Float
$f12	FP to print by syscall	-	FP to print by syscall
$s0	-	FxA (numerator of F)	-
$s1	-	FxB (denominator of F)	-
$s2	-	FxC (addition to F)	-
$s3	-	Result of F	-
$s4	-	-	-
$s5	-	-	-
$s6	-	A*A hold	-

4.0 | Learning Coverage:

4.1 | Learned ways to use one procedure, but use temporary values in it so that a different procedure is not required for testing with different valuables. Simply pass in the variables you wish to test each time.

4.2 | Learned how to use logical functions to determine if overflow is present in a calculation.

4.3 | Learned how to implement a multiplication into MIPS without using any built in multiplication functionality.

4.4 | Learned how to use incremental counters and branching functions to further the ability of programming in MIPS.

4.5 | Further improved my use of registers, using the same temporary register several times after the value in that register has been used and is now obsolete.

5.0 | Test Results

5.1 | Program 1 Test Results:

```
Welcome to Program 1!
Enter the first integer.
1
Enter the second integer.
2

The resulting value:
2.0
-- program is finished running --
```

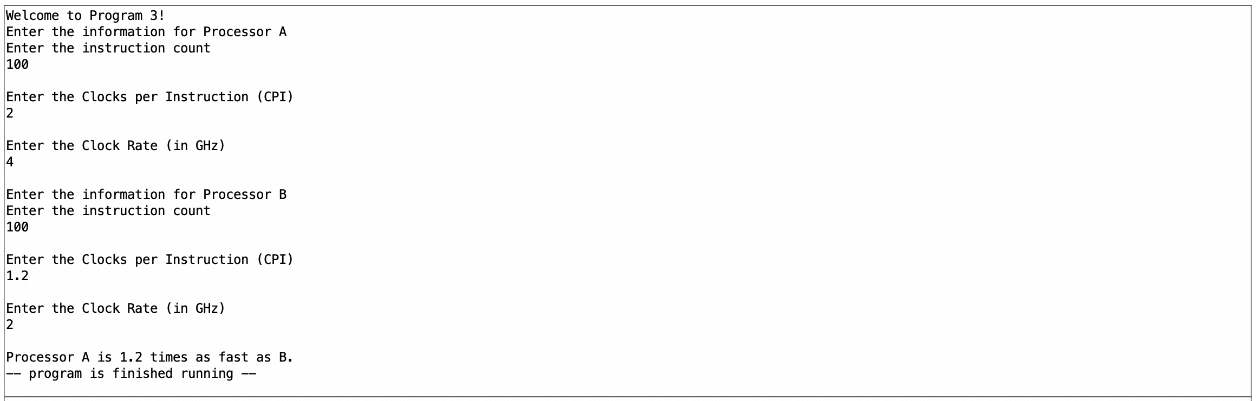
```
Welcome to Program 1!
Enter the first integer.
10
Enter the second integer.
5

The resulting value:
50.0
-- program is finished running --
```

5.2 | Program 2 Test Results:

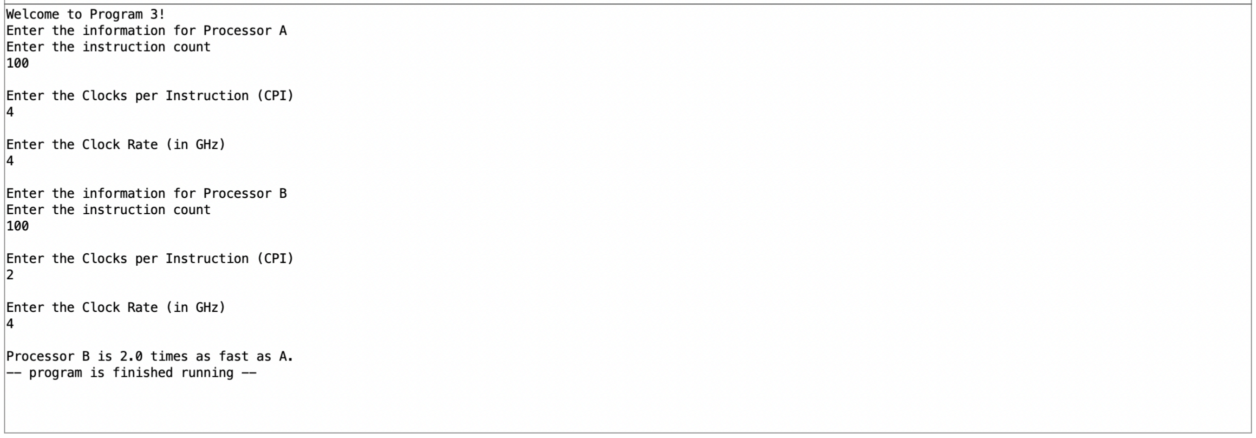
```
Welcome to Program 2!
Enter the value for variable a
0
Enter the value for variable b
2
Enter the value for variable c
2

No Overflow
Result of the function: 8
-- program is finished running (dropped off bottom) --
```

```
Welcome to Program 2!
Enter the value for variable a
-2
Enter the value for variable b
15
Enter the value for variable c
4

Divide by zero error
-- program is finished running --
```

5.3 | Program 3 Test Results:



```
Welcome to Program 3!
Enter the information for Processor A
Enter the instruction count
100

Enter the Clocks per Instruction (CPI)
2

Enter the Clock Rate (in GHz)
4

Enter the information for Processor B
Enter the instruction count
100

Enter the Clocks per Instruction (CPI)
1.2

Enter the Clock Rate (in GHz)
2

Processor A is 1.2 times as fast as B.
-- program is finished running --
```



```
Welcome to Program 3!
Enter the information for Processor A
Enter the instruction count
100

Enter the Clocks per Instruction (CPI)
4

Enter the Clock Rate (in GHz)
4

Enter the information for Processor B
Enter the instruction count
100

Enter the Clocks per Instruction (CPI)
2

Enter the Clock Rate (in GHz)
4

Processor B is 2.0 times as fast as A.
-- program is finished running --
```