Matthew Michael Sherlin
Dr. Augustine Samba
Computer Organization
November 6, 2020

Assembly Code File:

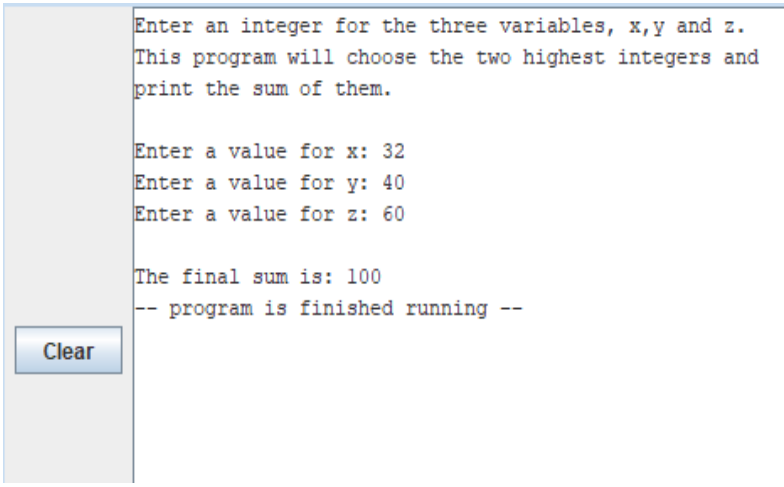
```
# Matthew Michael Sherlin
# Dr. Augustine Samba
# Computer Organization
# November 6, 2020

.data
        prompt1: .asciiz "Enter an integer for the three variables, x, y and z.\nThis program will choose the two highest integers and\nprint the sum of them."
        prompt2: .asciiz "\n\nEnter a value for x: "
        prompt3: .asciiz "Enter a value for y: "
        prompt4: .asciiz "Enter a value for z: "
        message1: .asciiz "\nThe final sum is: "

.text

li $v0, 4           #reading the first prompt
la $a0, prompt1
syscall

li $v0, 4           #reading the second prompt
la $a0, prompt2
syscall
li $v0, 5           #entering user input
syscall
move $s0, $v0   #moving user input into saved register

li $v0, 4           #reading the third prompt
la $a0, prompt3
syscall
li $v0, 5           #entering user input
syscall
move $s1, $v0   #moving user input into saved register


li $v0, 4           #reading the fourth prompt
la $a0, prompt4
syscall
li $v0, 5           #entering user input
syscall
move $s2, $v0   #moving user input into saved register
```

```
slt $t0, $s1, $s0          #compare y < x
beq $t0, $zero, else       #if false, go to else
add $t1, $t1, $s0          #add x to sum (temp1)
slt $t0, $s1, $s2          #compare y < z
beq $t0, $zero, else2      #if false, go to else
add $t1, $t1, $s2          #add z to sum
j endif
else2:
add $t1, $t1, $s1          #add y to sum
j endif

else:
add $t1, $t1, $s1          #add y to sum (temp1)
slt $t0, $s0, $s2          #compare x < z
beq $t0, $zero, else3      #if false, go to else
add $t1, $t1, $s2          #add z to sum
j endif
else3:
add $t1, $t1, $s0          #add x to sum
j endif

endif:

li $v0, 4                  #reading the final message
la $a0, message1
syscall

li $v0, 1                  #displaying total
move $a0, $t1
syscall

li $v0, 10                  #terminate program run and exit
syscall
```

Project Implementation:

In order to get this program to work, I first began by reading the initial prompts out and retrieving integers for the three variables used in the program. I used simple la and li instructions and move to achieve this basic part. Next, I had to create an algorithm to achieve the result desired. I began by using slt to compare x and y and create the if statement, then a beq instruction to jump to else if the if statement was false. In the case that the if statement was true, I added the x integer to a temporary register which I designated to be the sum. Next, I had to repeat the process to compare y to z and do the exact same thing with another else statement within the first if else statement. I basically repeated this process for each circumstance, using the same instructions just with the different results of the slt instruction. After I added the two largest integers, I jumped to the endif statement, and displayed the final message and sum. Again, I used basic la and li instructions to display these.

Working Code Screen Print:



Transcription:

Enter an integer for the three variables, x, y and z.
This program will choose the two highest integers and print the sum of them.

Enter a value for x: 32
Enter a value for y: 40
Enter a value for z: 60

The final sum is: 100
-- program is finished running --

Conclusion:

To conclude, the main lesson that I learned during this part of the lab was using a pen and paper externally in order to figure out my algorithm and keep track of my variables first, before going in and trying to write from scratch. At first, I was trying to write it, and I was getting confused trying to keep track of my variables and registers while I was writing the other sections of the code. So, because of this, I deleted what I had and tried to write it out by hand on paper to get it down. Finally, after I got it fleshed out on paper, I wrote it and it was much easier to do than before. In terms of issues faced, this part gave me way more than the first, however I learned that writing it out was much better. I will definitely do this when writing more difficult code in the future. Big learning lesson here for me. Besides that, I really enjoyed writing this code again and I am enjoying writing in MIPS.