1 Project 2: MIPS Assembly Code Programming Using MARS Tool

10 points

In the previous project (Project 1), you were introduced to the MARS tool. For this project, you will use the MARS tool to develop and run a MIPS assembly language program.

1. Create and run MIPS assembly code for the following high-level language (HLL) code. I have provided both C++ and Python versions of the HLL code.

C++ Version	**Python Version**

`int j = 10;`
`int x = 1;`
 	
`do`
`{`
`  x = x + 3*j;`
`  cout << x << " ";`
`  j--;`
`} while (j > 0);`
 	
`cout << endl;`
	`j = 10`
	`x = 1`
	
	`while True:`
	`  x = x + 3*j`
	`  print(x,end=" ")`
	`  j -= 1`
	`  if (j == 0):`
	`  break`

2. Save the program as an *.asm file – use the first initial of your first name and the first 4 letters of your last name. For example, my file submission name would be **tmani.asm**.
3. **Take a screen shot of the MARS console (including Run/IO section) so that the grader can view your results.** If you take a screen shot of the entire console, please crop and enlarge the Run/IO section so that it is readable.
4. **Submit the actual *.asm file so that the grader can run it.**
5. Turn in your code and output (screen shot) for credit. **Make sure that your name appears on <u>both</u> documents that you submit so that you can get proper credit for your work.**

2 SOLUTION

2.1 Code (.asm)

An example of the MIPS assembly code solution is the following:

```
################################
# tmani.asm
# Theodore Manikas
# 2022 Nov 29
# CS/ECE 5/7381 Spring 2023
################################

.data
j:          .word 10                #initial values for variables j and x
x:          .word 1
three:      .word 3                 #multiplier constant
space:      .asciiz  " "          # space to insert between numbers


.text
           lw    $s0, j            #load variables into registers
           lw    $s1, x
           lw     $s2, three

loop:       mul $s3,$s2,$s0         # 3*j
            add $s1,$s1,$s3         # x = x + 3*j
            li    $v0,1             # print x (integer value)
            add $a0, $s1, $zero
            syscall
            la    $a0, space      #load address of spacer for syscall
            li    $v0, 4          # specify Print String service
            syscall               # print the spacer string
            subi $s0,$s0,1            # j-- (j=j-1)
            bgtz $s0,loop             # repeat loop if j>0
            li    $v0, 10         # system call for exit
            syscall               # we are out of here.
```

2.2 Results