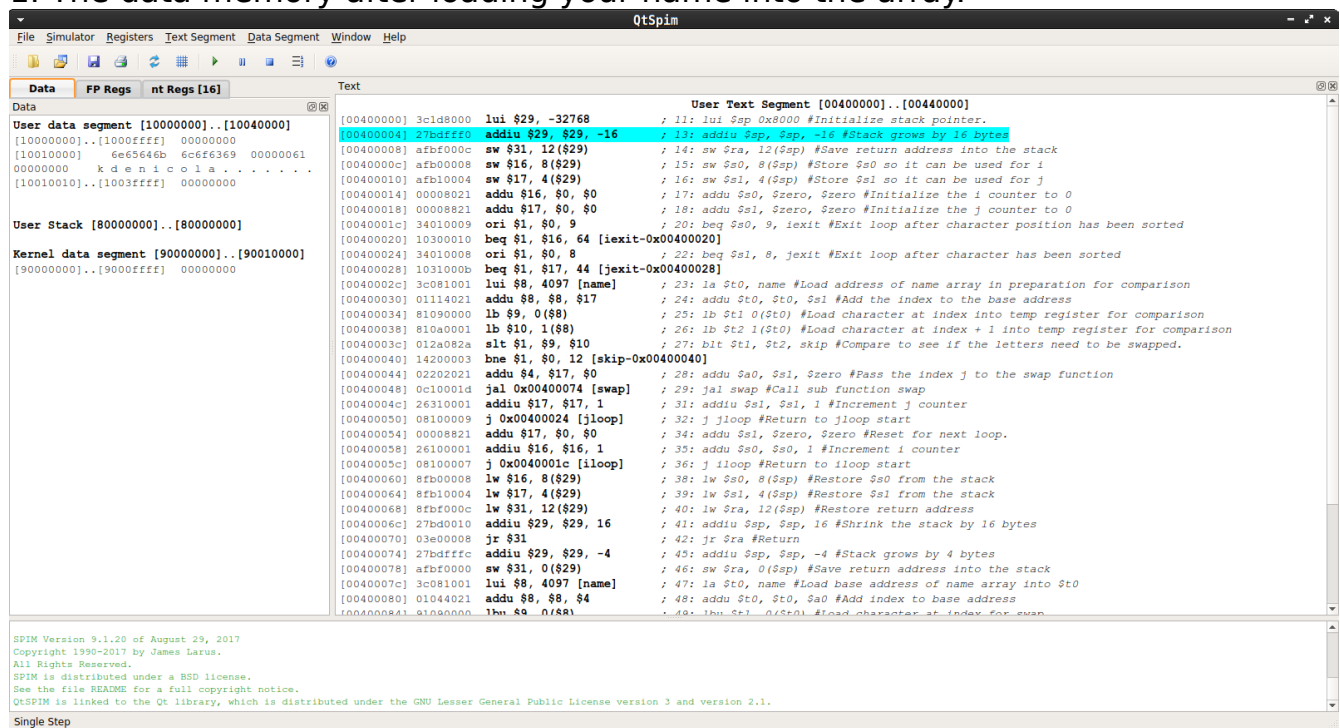


Describe in one paragraph some of the difficulties you encountered while writing the program

I started the project by working on the main function. Initially, I thought it was possible to use variables as the constant values for accessing the array (you would think that the word 'constant' would have tipped me off), but realized before I got to testing that I needed to adjust how I was accessing the array. I recalled information from lecture 1 about accessing arrays and fixed the issue by using constant values with a base address. Once I ran my first test, I was perplexed by a bad address error until I realized I had mixed up the argument register with the value return register. My last major issue was that I was multiplying the index by 4, forgetting that I was accessing bytes and not words. I adjusted for this, and it resolved the issues of indexing outside the array.

Please include the following screenshots taken after your program is working:

1. The data memory after loading your name into the array.



The screenshot shows the QtSpim MIPS simulator interface. The 'Data' tab is selected, displaying the memory layout. The 'User data segment' is located at address 10000000 and contains the string 'k d e n i c o l a . . . . .'. The 'User Stack' is located at address 80000000 and is empty. The 'Kernel data segment' is located at address 90000000 and is empty. The 'Text' tab is also visible, showing the assembly code for the program. The code includes instructions for loading the stack pointer, initializing the stack, and performing a bubble sort on the array 'name'.

```
[00400000] 3c1d8000 lui $29, -32768 ; 11: lui $sp, 0x8000 #Initialize stack pointer.
[00400004] 27bdf000 addiu $29, $29, -16 ; 13: addiu $sp, $sp, -16 #Stack grows by 16 bytes
[00400008] afbf000c sw $31, 12($29) ; 14: sw $ra, 12($sp) #Save return address into the stack
[0040000c] afbf0008 sw $16, 8($29) ; 15: sw $s0, 8($sp) #Store $s0 so it can be used for i
[00400010] afbf0004 sw $17, 4($29) ; 16: sw $s1, 4($sp) #Store $s1 so it can be used for j
[00400014] 00008021 addu $16, $0, $0 ; 17: addu $s0, $zero, $zero #Initialize the i counter to 0
[00400018] 00008821 addu $17, $0, $0 ; 18: addu $s1, $zero, $zero #Initialize the j counter to 0
[0040001c] 34010009 ori $1, $0, 9 ; 20: beq $s0, 9, iexit #Exit loop after character position has been sorted
[00400020] 10300010 beq $1, $16, 64 [iexit-0x00400020]
[00400024] 34010008 ori $1, $0, 8 ; 22: beq $s1, 8, jexit #Exit loop after character has been sorted
[00400028] 1031000b beq $1, $17, 44 [jexit-0x00400028]
[0040002c] 3c081001 lui $8, 4097 [name] ; 23: la $t0, name #Load address of name array in preparation for comparison
[00400030] 01114021 addu $8, $8, $17 ; 24: addu $t0, $t0, $s1 #Add the index to the base address
[00400034] 81090000 lb $9, 0($8) ; 25: lb $t1, 0($t0) #Load character at index into temp register for comparison
[00400038] 810a0001 lb $10, 1($8) ; 26: lb $t2, 1($t0) #Load character at index + 1 into temp register for comparison
[0040003c] 012a082a slt $1, $9, $10 ; 27: blt $t1, $t2, skip #Compare to see if the letters need to be swapped.
[00400040] 14200003 bne $1, $0, 12 [skip-0x00400040]
[00400044] 02202021 addu $4, $4, $17, $0 ; 28: addu $a0, $s1, $zero #Pass the index j to the swap function
[00400048] 0c10001d jal 0x00400074 [swap] ; 29: jal swap #Call sub function swap
[0040004c] 26310001 addiu $17, $17, 1 ; 31: addiu $s1, $s1, 1 #Increment j counter
[00400050] 08100009 j 0x00400024 [jloop] ; 32: j jloop #Return to jloop start
[00400054] 00008821 addu $17, $0, $0 ; 34: addu $s1, $zero, $zero #Reset for next loop.
[00400058] 26100001 addiu $16, $16, 1 ; 35: addu $s0, $s0, 1 #Increment i counter
[0040005c] 08100007 j 0x0040001c [iloop] ; 36: j iloop #Return to iloop start
[00400060] 8fb00008 lw $16, 8($29) ; 38: lw $s0, 8($sp) #Restore $s0 from the stack
[00400064] 8fb10004 lw $17, 4($29) ; 39: lw $s1, 4($sp) #Restore $s1 from the stack
[00400068] 8fbf000c lw $31, 12($29) ; 40: lw $ra, 12($sp) #Restore return address
[0040006c] 27bd0010 addiu $29, $29, 16 ; 41: addiu $sp, $sp, 16 #Shrink the stack by 16 bytes
[00400070] 03e00008 jr $31 ; 42: jr $ra #Return
[00400074] 27bdf000 addiu $29, $29, -4 ; 43: addiu $sp, $sp, -4 #Stack grows by 4 bytes
[00400078] afbf0000 sw $31, 0($29) ; 46: sw $ra, 0($sp) #Save return address into the stack
[0040007c] 3c081001 lui $8, 4097 [name] ; 47: la $t0, name #Load base address of name array into $t0
[00400080] 01044021 addu $8, $8, $4 ; 48: addu $t0, $t0, $a0 #Add index to base address
[00400084] 810a0000 lb $9, 0($8) ; 49: lb $t1, 0($t0) #Load character at index for swap
```

Counting the number of lines in QT spim yielded a total of 40. Counting the number of instructions in the text document yielded 37.