CSI 3640: Computer Organization
Fall 2024, OAKLAND UNIVERSITY
MIPS Programming Assignment #2
Due: Tuesday, November 5, 2024, 11:59PM
100 points, amounts to 10% of the final grade.

In this final MIPS programming assignment, you will implement user-defined procedures by manipulating the stack segment of a process.

You are asked to implement a MIPS procedure named string_insert. The procedure will take two null-terminated ASCII strings: str1 and str2 and an integer position value pos as input parameters. When invoked, the string_insert procedure will insert a copy of str2 into str1 right before the index location indicated by pos. For example, if str1 contains the string "ABCDEFG" and str2 contains "123", then a procedure call string_insert with the pos parameter value of 5 would update str1 into "ABCDE123FG"; since the letter F had the index location of 5. If the pos value exceeds the lengths of str1, then str2 should be concatenated at the end of str1. Of the pos value equals to 0; then str2 will be prepended to str1. The string_insert procedure will not return any value.

Furthermore, you also need to implement a second MIPS procedure named strlen. Like the C library function of the same name, strlen will take a string as input parameter, and return the number of characters in that string, excluding the null character. The return value of strlen will be available at the $v0 register for the caller procedure to read.

You should try to implement strlen first, and then focus on string_insert because the implementation of substring_count can be greatly simplified when it can invoke the strlen procedure.

The main function of the program is given in the following. You may define any additional number of helper procedures as necessary. However, the data segment as well as the body of the main function should be used exactly as given. If you need space to temporarily hold bulks of data, you can and should utilize the stack segment of the process.

```
.data

#Static arrays used to store the two string inputs

str1: .space 200 # reserve a 200-byte memory block
str2: .space 200 # reserve a 200-byte memory block

#String literals

printstr1: .asciiz "Enter the first string: "
printstr2: .asciiz "Enter the second substring: "
printstr3: .asciiz "Enter the insertion position: "
printstr4: .asciiz "After insertion, updated first string is: "

.text
.globl main

main:
  li, $v0, 4 #to print prompt#1
  la $a0, printstr1
  syscall

  li, $v0, 8 #input the first string
  la $a0, str1
  li $a1, 200
  syscall

  li, $v0, 4 #print prompt #2
  la $a0, printstr2
  syscall

  li, $v0, 8 #input the second string
  la $a0, str2
  li $a1, 200
  syscall

  li, $v0, 4 #to print prompt#3
  la $a0, printstr3
  syscall

  li, $v0, 5 #input the position value(integer)
  syscall #pos stored in $v0

  la $a0, str1 #load the address of str1 to $a0
  la $a1, str2 #load the address of substr1 to $a1
  add $a2, $0, $v0 # load the position value to $a2`
  jal string_insert #procedure call from main
```

```
  li, $v0, 4 #print string mode

  la $a0, printstr4 #Literal part of Output
  syscall
  la $a0, str1 # load address of str1 for output
  syscall

  li, $v0, 10 #clean exit
  syscall

string_insert:
    #definition of string_insert goes below

    # Any additional function definition(s), including strlren
```

To get full credit for your work, your MIPS code must be sufficiently documented with single line comments explaining the purpose of the key parts of the program.

Save your MIPS program in a single plaintext file and submit it to Moodle. Good Luck!