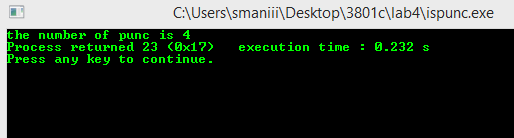
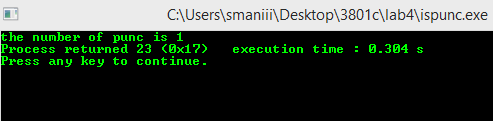
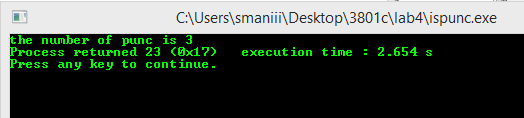
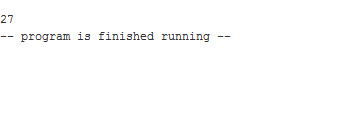
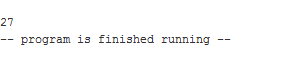
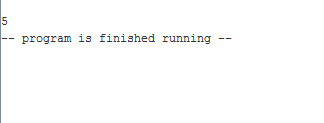
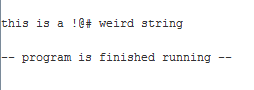
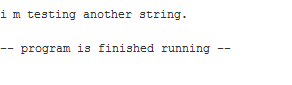
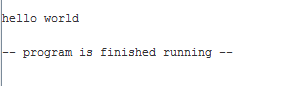
Lab Report 4 Cover Page Portfolio 3

1. Project Description:
   1. This project is about working and manipulating strings. For the first part we find how many punctuations are in a string. The second works on how many characters are in a string in MIPS assembly. The third is about converting Uppercase letters to lower case letters in MIPS assembly. The input for all of these projects is a hard coded string. Each program has a different output. The first the number of punctuations, the second is the number of characters, and the third is a string with the uppercase letter converted to lower case.
2. Solution to Problems and Assembly Coding Basics:
   1. Study set 5 exercise:
      1. C. SpecRatio(A)=Time(R)/Time(A)
         1. A SpecRatio is defined as the time of reference over the time of test machine.
      2. 8/6
         1. GraphicSpecRatio(B)=GraphicsTime(R)/GraphicsTime(B)
      3. The graphics performance of machine b is 1.33 fold of the reference machine.
      4. C-Prototype of number of punctuation.
         1. #include <stdio.h>
         2. #include <string.h>
         3. main() {
         4. char hi[50] = "This is a test. 1 2 3 4 5 !!"; //create a string
         5. int i;//declare an int for the loop
         6. int x=0; //declare an int for the count
         7. for (i=0;i<40;i++) { //for loop statement
         8. if (ispunct(hi[i]) != 0) { //if it a punctuation
         9. x++;//add one to count
         10. }
         11. }
         12. printf("the number of punc is %d",x);//print the count.
         13. }
      5. Find the string length
         1. .data
         2. str: .asciiz "123456789"
         3. .text
         4. # Load address of string into a0
         5. la $t2, str #load the string on a reg
         6. strlen:
         7. li $t0, 0 # initialize the count to zero
         8. loop:
         9. lb $t1, 0($t2) # load the next character into t1
         10. beqz $t1, exit # check for the null character
         11. addi $t0,$t0,1 #add one to count
         12. addi $t2, $t2, 1 #add one to reg number
         13. j loop #jump back to loop
         14. exit:#exit label if condition is true
         15. li $v0, 1 #load reg for int output
         16. move $a0, $t0 #move to output reg
         17. syscall
         18. li $v0, 10#load reg for close program
         19. Syscall
             1. First save the data string
             2. Create your loop statement
             3. End if the null character is shown.
             4. Else add one to count and register address
             5. Output the number of characters.
             6. End the program
      6. To lowercase
         1. .data
         2. string: .asciiz "THis is a STRING with Both Upper, LOWER space and Special !@# to an ALL LOWER striNG" # We want to lower this string
         3. newline: .asciiz "\n"
         4. .text
         5. main:
         6. la $t0, string # Load here the string
         7. toLowerCase:
         8. lb $t2, 0($t0) # We do as always, get the first byte pointed by the address
         9. beqz $t2, end # if is equal to zero, the string is terminated
         10. #if (character >= 'A'
         11. blt $t2, 65, continue
         12. upperCaseTest2:
         13. # && character <= 'Z')
         14. blt $t2, 90, isUpperCase
         15. continue:
         16. # Continue the iteration
         17. addi $t0, $t0, 1 # Increment the address
         18. j toLowerCase
         19. isUpperCase:
         20. # add 32, so it goes lower case
         21. addi $t2,$t2,32
         22. sb $t2, 0($t0) # store it in the string
         23. j continue # continue iteration as always
         24. end:
         25. li $v0, 4 # Print the string
         26. la $a0, string
         27. syscall
         28. li $v0, 4 # A nice newline
         29. la $a0, newline
         30. syscall
         31. # We have done, exit the program
         32. li $v0, 10
         33. syscall
             1. Save string data
             2. Load string into register
             3. Find if the null character is shown.
             4. Else find the character is uppercase
             5. If is upper change to lower case by increasing the asci number by 32.
             6. Else move to next character.
             7. Print out the new array.
             8. Close program.
3. Learning coverage
   1. Understanding the inpunc function in C to find if a character is a punctuation symbol.
   2. Understanding how to manipulate a string by changing the asci number.
   3. Understanding how to look at a string character by character.
   4. Understanding how to write a counter of characters in MIPS assembly.
   5. Understanding how to read and use the asci chart.
4. Prototype in C-language:
   1. Look at section 2. iv.
   2. String length
      1. #include <stdio.h>
      2. #include <string.h>
      3. main() {
      4. char hi[50] = "This is a test. 1 2 3 4 5 !!";
      5. int i;
      6. int x=0;
      7. x = strlen(hi);
      8. printf("the number of char is %d",x);
      9. }
   3. To Lowercase
      1. #include <stdio.h>
      2. #include <string.h>
      3. main() {
      4. char hi[50] = "This is a test. 1 2 3 4 5 !!";
      5. int i;
      6. int x=0;
      7. for(i = 0; i<40; i++){
      8. hi[i] = tolower(hi[i]);
      9. }
      10. printf("%s",hi);
      11. }
5. Test Plan:
   1. The first program tests are the base case of "This is a test. 1 2 3 4 5 !!", “I’m testing another string.”, and “ThiS is A !@# Weird StrinG”. This provides adequate test coverage because it tests a base case, a normal case, and an extreme case. 
   2. The second program tests are the base case of " abcde", “I’m testing another string.”, and “ThiS is A !@# Weird StrinG”. This provides adequate test coverage because it tests a base case, a normal case, and an extreme case.
   3. The third program tests are the base case of " HeLlo WoRld", “I’m testing another string.”, and “ThiS is A !@# Weird StrinG”. This provides adequate test coverage because it tests a base case, a normal case, and an extreme case. 
6. Reference list
   1. For this report, the portfolio file was used a reference sheet. It has the questions given and hints on how to program the MIPS code.
   2. The report grading rubric gives advice on how to write the lab report.