“String operations”

-This Program were made to introduce 6 operations on string:

-String Length.

-Number of repetition of character .

-String Reverse.

-IndexOf the Character.

-SubString.

-Change To Uppercase.

Main:

```
Main:
# loop to begin system
li $v0 , 4
la $a0 , msg
syscall       # print -> Enter String:
li $v0 , 8
la $a0 , str
la $a1 , 20
syscall  #input string
do:
li $v0 , 4
la $a0 , start
syscall  #print message -> "Hello, Enter number from 1 t0 6 for string operation or 7 t0 exit"
li $v0 , 4
la $a0 , choices
syscall   #print message to visiualize choices of operation
          #1- string length  2- search  3-reverse  4- index of  5- substring  6- UpperCase
li $v0 , 5
syscall    # user input choice
beq $v0 , 1 , p1
beq $v0 , 2 , p2
beq $v0 , 3 , p3
beq $v0 , 4 , p4
beq $v0 , 5 , p5
beq $v0 , 6 , p6
beq $v0 , 7 , p7

li  $v0  ,  4
la  $a0  ,  Wrong
syscall    #print  ->   "Wrong choice Try Again!"
j  do          #jump  to  do
```

If user input 1:

```
############### find length ######################
p1:
la $a0 , str  #load string address
jal strLength  # jump to finction
li $v0 ,4
la $a0 , msg_length
syscall  #print -> length of string
li $v0 , 1
move $a0 , $v1 #move length in $a0
syscall #print lenth
j do #jump to do
```

-First we are going to explain the length function:

This function will count the number of characters in the string.

-The C code:

```
int lengthStr (string str)
{
      int i=0, length = 0;
      while(str[i]!='\0')
      {
            length++;
            i++;

      }
      return length;
}
```

-Input: hello people

-Output will be:

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
1


length of string = 12
```

-The Assembly code:

```
##### $a0 -> string
strLength:
# save register $s0 , $s1 in stack
addi $sp , $sp , -8
sw $s0 , 0($sp)
sw $s1 , 4($sp)

add $s0 , $0 ,$0    #i=0  intialize index of string
add $s1 , $0 , $0  #length = 0  intialize variable to save length

while1:               #loop
add $t0 , $s0 , $a0       #str1[i]
lbu $t1, 0($t0)           # load character in str[i] in $t1
beqz $t1, exit1           # check if str[i] == '\0' if $t1 = 0 -> exit , else
addi $s0 , $s0 ,1         # i++
addi $s1, $s1, 1          # length ++
j while1
 exit1:    #exit if condition true
add $v1 , $s1 , -1   #retrun length
# load values from stack
lw $s0 , 0($sp)
lw $s1 , 4($sp)
addi $sp , $sp , 8
jr $ra    #end
```

Return to main, print length then jump to do which introduce list of operations.

If user input 2:

```
p2:
li $v0 , 4
la $a0 , msg_search2 #print -> charcter:
syscall
li $v0 , 12
syscall  #user input character
move $a1 , $v0  #save character in $a1
la $a0, str     #save add. of string $a0
jal search  #jump to func.
li $v0 , 4
la $a0 , msg_search
syscall  #print found:
#print number 0 : not found , any number : found and number indicate to The number of repetitions
li $v0 , 1
move $a0 , $v1
syscall #print number of repetition
j do  #jump to do
```

-Second we are going to explain the Search function:

This function will Search about a character in the string then print the number of the times this character will appear in the string.

-The C code:

```
int Search (string str , char c)
{    int i = 0,found = 0;
    for (i=0;i<lengthStr(str);i++)
    {
        if(c == str[i])
        {
            found++;
        }
    }
    if(found == 0 )
     return 0;
    else{
        return found;
    }
}
```

when the string is “hello people”.

-Input: character “o”.

-Output will be: found = 2

number indicate to number of the times this character will appear in the string.

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
2

charcter:o
found: 2
```

-The Assembly code:

```
########## procedure of search #################
#$a1 = character , $a0 = string
search:
# save value in $ra in stack
addi $sp , $sp , -4
sw $ra , 0($sp)

add $s0 , $0 ,$0   # i=0 , intialize index of string
add $s1 , $0 , $0  # founde =0 ; intialize variable indicato to number of repetition of char in string
jal strLength      # call procedure to know length
for1:
slt $t0 , $s0 , $v1   # if i < length -> $t0 =1 , else  -> $t0=0
beqz $t0 , exit2      # if $t0 =0 -> exit , else
add $t0 , $s0 , $a0   #str[i]
lbu $t1 , 0($t0)      #load character
addi $s0 , $s0 ,1     # i++
bne $a1 , $t1 , for1   #if ch != str[i] -> loop
addi $s1 , $s1 , 1     #found++
j for1       # jump untill i > length

exit2:
move $v1 , $s1        # return 0 -> indicat to this char not found in string
lw $ra , 0($sp)
addi $sp , $sp , 4
jr $ra  #end
```

Return to main, print number of repetition then jump to do which introduce list of operations.

If user input 3:

```
p3:
li $v0 , 4
la $a0 , msg_revers
syscall  #print Reverse String:
la $a0 , str
move $a1 , $a0 #save string add. in $a0
jal reverse  #jump to func.
j do  #jump to do
```

-Third we are going to explain the String Reverse:

This function will Reverse the Input string then print The String from the Length to 0 index.

-The C code:

```
void reverseString(string str , int n)
{    int i =0;
     while (str[n-1] != '\0')
     {

          cout<<str[n-1];
          n--;
     }
}
```

-Input : hello.

-Output will be : olleh.

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
3

Reverse String: olleh
```

-The Assembly code:

```
###################### reverse procedure ################ ###############
#### $a1 -> string
reverse:
addi $sp , $sp , -8
sw  $ra , 0($sp)
jal strLength      # call procedure to know length of string $v1 -> length
While2:      #loop
addi $t0 , $v1 ,-1  # length-- ; save it in $t2
add $t1 , $a1 , $t0  #str[length-1] -> last char in string
lbu $t2 , 0($t1)     # load char in $t3
beqz $t2 , exit3    # if char = 0 -> exit  "End of string"
li $v0 , 11          # load in $v0 , 11 to print char
la $a0 , 0($t2)      # load add of char
syscall              #print
addi $v1 , $v1 , -1  #length-=length -> change length
j While2  # jump to loop until end
exit3:
lw $ra , 0($sp)
addi $sp , $sp ,8
jr $ra #end
```

Return to main then jump to do which introduce list of operations.

If user input 4:

```
############### UpperCase #################
p4:
la $a0 , str
move $a1 , $a0 #load add. of string in $a0
jal UpperCase  #jump to func
j do           # jump to do
```

-fourth we are going to explain Change To Uppercase Function.

-this function will get the String and return it in the Form of Uppercase.

-The c code:

```
void UpperCase (string str , int length)
{
     int i;
     for(i=0;i<lengthStr(str);i++)
     {
         if (str[i] == 32)
            cout<<str[i];
         if (str[i] < 97)
              cout<<str[i];

         if (str[i] >= 97)
           {
                str[i]=str[i]-32;
                cout<<str[i];
           }
     }

 }
```

-Input :“hello people”.

-Output will be: HELLO PEOPLE.

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
4
HELLO PEOPLE
```

-The Assembly code:

```
################## UPPerCase #################
##### $a1 -> string
UpperCase:    # $a1 -> string
addi $sp $sp ,-4
sw $ra , 0($sp)

jal strLength      #call length procedur to know length , $v1 = length
add $s0 , $0 ,$0   # i=0 , initialize index
for3: #loop
bgt $s0 , $v1 , exit51  # i < length $t0=1 , i>length $t0 =      # if $t0==0 -> exit
add $t0 , $s0 ,$a1       # $t0==1 , str[i]
lbu $t1 , 0($t0)     # load char
beq $t1 , 32 , space
slti $t2 , $t1 , 97     # str[i] < 32 -> capital ,$t2=1  notbranch ,  $t2= 0 ,branch -> small
bnez $t2 , else4      #if $t2 != 0 -> else , capital
addi $t1 , $t1 ,-32   #ascii code for char - 32 -> char in uppercase
print2:  #print
li $v0 , 11
la $a0 ,0($t1)
syscall  #print char
addi $s0 , $s0 , 1  #i++
j for3
else4:  # $t2 !=0 , char in upper
  j print2
 space:
 j print2
exit51:
lw $ra , 0($sp)
addi $sp ,$sp ,4
jr $ra  #end
```

Return to main then jump to do which introduce list of operations.

If user input 5:

```
########### indexOf ################
p5:
li $v0 , 4
la $a0 , msg_indexC
syscall  #print -> Enter Character:
li $v0 , 12
syscall    #user input character
move $a2 , $v0   #save char. in $a2
li $v0 , 4
la $a0 , msg_indexP
syscall  #print -> Enter Start Position:
li $v0 , 5
syscall   #user input start position
move $a3 , $v0  #save start pos. in $a3
la $a0 , str #load add. string $a0
jal indexOf  #jumpp to func.
li $v0 , 4
la $a0 ,  msg_index
syscall  #print -> index of characte:
li $v0 , 1
move $a0 , $v1
syscall  #print index
j do #ju,p to do
```

-Fifth we are going to explain IndexOf the Character.

This function will get the character and start position for searching and return its index.

-if the start position is greater than length of the string the function will return -1 .

-if the start position is less than zero the function will start from zero.

-if the character is not found the function will return -1.

-The C code:

```
int indexOf (string str , char c,int start)
{
     int i, id =0 ;
     if(start >lengthStr(str))
          return -1;
   else {
     if (start < 0)
        start = 0;

     for(i=start ;i<lengthStr(str);i++)
     {
          if(c == str[i])
          {   id++;
               break;
          }
     }
    if (id == 0)
     return -1;
    else
     return i;}
}
```

input String : “hello people”

-Input: character “o” ,start =0.

-Output will be: index = 4

number indicate to index of character.

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
5

Enter Character:o
Enter Start Position: 0

index of characte:4
```

Case : start < 0 “-ve value”

-Input: character “o” ,start =-3.

-Output will be: index = 4

number indicate to index of character.

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
5

Enter Character:o
Enter Start Position: -3

index of characte:4
```

Case: character not found

-Input: character “z” ,start =0.

-Output will be: index = -1

number indicate to not found

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
5

Enter Character:z
Enter Start Position: 0
```

Case: start > length

-Input: character “o” ,start =23.

-Output will be: index = -1

number indicate to out of range

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
5

Enter Character:o
Enter Start Position: 23

index of characte:-1
```

-The Assembly code:

```
################### indexOf ####################
#### $a3 -> start position , $a2-> character , $a0 -> string
indexOf:
addi $sp , $sp , -4
sw $ra , 0($sp)

add $s1 , $0 , $0        #i=0 ; intialize index
jal strLength            # call procedure to knoe length , $v1 -> length
blt  $a3 , $v1 , else2    # posi < length -> else
addi $v1 , $0 , -1         #  return -1 "out of range"
lw $ra , 0($sp)
addi $sp ,$sp ,4
jr $ra               #end procedure
else2:
bgtz  $a3 , case       #posi >  0 -> for
case:
move $s0 , $a3
j for2
move $s0 , $0           #pos < 0 -> pos =0
for2:              ### begining of search
bgt $s0 , $v1 , exit4 # pos > length -> exit
add $t0 , $a0 , $s0    #str[i]
lbu $t1 , 0($t0)       #load char in $t1
addi $s0 , $s0 ,1      # pos++
bne $a2 , $t1 , for2    # chaeck if char != str[i] , jump to loop
addi $s1 , $s1 , 1      # if char == str[i] ,id++
addi $s0 , $s0 ,-1      # pos--;
exit4:
bne $s1 , $0 , else3    #if id !=0 -> else
addi $v1 , $0 , -1          # i==0 -> return value =-1 "not found"
lw $ra , 0($sp)
addi $sp , $sp ,4
jr $ra  #end
else3:  # i= number
move $v1 , $s0 #return number
lw $ra , 0($sp)
addi $sp , $sp ,4
jr $ra #end
```

Return to main , print index of character then jump to do which introduce list of operations.

If user input 6:

```
################ SubString ################
p6:
li $v0 ,4
la $a0 , msg_SUBI
syscall #print -> Enter intial index:
li $v0 , 5
syscall    #user input intial index
move $a1 , $v0  #save index in $a1
li $v0 ,4
la $a0 , msg_SUBF
syscall #print -> Enter end index:
li $v0 , 5
syscall #user input final index
move $a2 , $v0  #save index in $a2
la $a0 , str
move $a3 , $a0  #save add. string in $a3
jal subString #jump to func
li $v0 , 4
la $a0 , newline
syscall  #print newline
j do #jump to do
```

-sixth we are going to explain substring function.

-this function will get the start index and end index to cut substring from string.

The c code:

```
void SubString (string str , int in , int fin)
{
    if (in < 0 || fin < in || fin > lengthStr(str) )
        cout<<"Exception Error";
   else {

    while(in != fin)
    {
        cout<<str[in];
        in++;
    }
}
}
```

Input string : “hello people”

-Input : start = 5 , end = 12.

-Output will be: people .

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
6

Enter intial index:5
Enter end index:12
 people
```

Case : start > end

-Input : start = 12 , end = 5.

-Output will be: Exception Error

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
6

Enter intial index:12
Enter end index:5

Exception Error
```

case: start < 0

-Input : start = -1 , end = 12.

-Output will be: Exception Error

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
6

Enter intial index:-1
Enter end index:12

Exception Error
```

Case: end > length

-Input : start = 1 , end = 19.

-Output will be: Exception Error .

```
Hello, Enter number from 1 t0 6 for string operation or 7 t0 exitt
 1- string length
 2- search
 3-reverse
 4- UpperCase
 5- index of
 6- substring
6

Enter intial index:1
Enter end index:19

Exeception Error
```

-The Assembly code:

```
############### SubString ######################
#### $a1 -> start , $a2 -> end , $a3 -> string
subString:
addi $sp , $sp , -4
sw $ra , 0($sp)
#case1
bltz  $a1 , exit5        # if start < 0 ,branch exit5
#case2
blt  $a2 , $a1 , exit5   #if end < start ,branch exit5
jal strLength            ###length=v1
#case3
bgt  $a2 , $v1, exit5    #if end > length,branch exit5
while3:                  #loop
beq $a1 , $a2 , exit6    # if start == end -> exit6
print:                   #print char
add $t0 , $a1 , $a3      #str[start]
lbu $t1 , 0($t0)         #load char in t1
li $v0 , 11              #print char
la $a0 ,0($t1)
syscall
addi $a1 , $a1 , 1       #start++
j while3 #jump to loop until start == end
exit5:
li $v0 , 4
la $a0 , msg_SUBE        ##msg_SUBE: .asciiz "\nExeception Error"
```

```
syscall
lw $ra , 0($sp)
addi $sp , $sp ,4
jr $ra   # end and go to main
exit6:
lw $ra , 0($sp)
addi $sp , $sp ,4
jr $ra   # end and go to main
```

Return to main then jump to do which introduce list of operations.

If user input 7:

```
############ End Program #############
p7:
j EndProgram
EndProgram:
li $v0 , 10
syscall
```