COL216 Lab Assignment-1 (S1)

- Mayank Mangla (2020CS50430)

1 Introduction

This project's aim is to develop a program in ARM Assembly Language for arranging a sequence of character strings in dictionary order (lexicographic order).
Stage 1: This program compares two input strings and outputs whether the first string is less than, greater than, or equal to the second string.
The report for the stage 1 program is given here:

2 Program

(i) The first part of the code manages all the commands and directs to the next major instruction. It first prints the initial string which asks the user for the input i.e., two strings and one comparison mode. Then it loads the strings to different registers.

```
manager:
    ldr r1,=PRI_space
    ldr r2,=initial_str
    str r2,[r1,#4]
    mov r0,#0x05
    swi 0x123456
    ldr r2,=empty
    mov r4,r2
    bl read_input
    mov r5,r2
    bl read_input
    mov r6,r2
    bl read_input
    ldrb r1,[r6],#1
    cmp r1,#49
    beq comparator_1
    bl comparator_2
```

From this the strings goes for comparison on the basis of comparison mode entered.

(ii) This is a loop for comparison in case sensitive mode. It breaks only if any one character of the first string is unequal to the corresponding character of the second string or both the strings terminated. In case of termination of both the strings the input strings are equal and then we output the strings to be equal. On the other hand, if we get that first string is less than or greater than the second string then we jump on the label that outputs the corresponding result.

```
comparator_1:
    ldrb r2,[r4],#1
    ldrb r3,[r5],#1
    cmp r2,r3
    blt one_lt_two
    bgt one_gt_two
    cmp r2,#0x00
    beq one_eq_two
    bl comparator_1
```

(iii) The third part is the comparing loop in case of case in-sensitive comparison. This check for the ASCII value of the character and if it is greater than 91 that implies the character is a small English alphabet and hence subtracts the number 32 from its ASCII code to make it in CAPITAL. Then the comparison is made keeping both the characters as capitals. Rest all is the same from comparator1.

```
comparator_2:
    ldrb r2,[r4],#1
    ldrb r3,[r5],#1
    cmp r3,#91
    bgt moderate_1
m1:    cmp r2,#91
    bgt moderate_2
m2:    cmp r2,r3
    blt one_lt_two
    bgt one_gt_two
    cmp r2,#0x00
    beq one_eq_two
    bl comparator_2
moderate_1:
    sub r3,r3,#32
    bl m1
moderate_2:
    sub r2,r2,#32
    bl m2
```

(iv) These labels are used to print the corresponding result and exit the program

```
one_lt_two:
    ldr r1,=words
    ldr r2,=less_than
    str r2,[r1,#4]
    mov r3,#38
    str r3 ,[r1,#8]
    mov r0,#0x05
    swi 0x123456
    mov r0, #0x18
    swi 0x123456
one_gt_two:
    ldr r1,=words
    ldr r2,=greater_than
    str r2,[r1,#4]
    mov r2,#41
    str r2 ,[r1,#8 ]
    mov r0,#0x05
    swi 0x123456
    mov r0, #0x18
    swi 0x123456
one_eq_two:
    ldr r1,=words
    ldr r2,=equal_to
    str r2,[r1,#4]
    mov r2,#35
    str r2 ,[r1,#8]
    mov r0,#0x05
    swi 0x123456
    mov r0, #0x18
    swi 0x123456
```

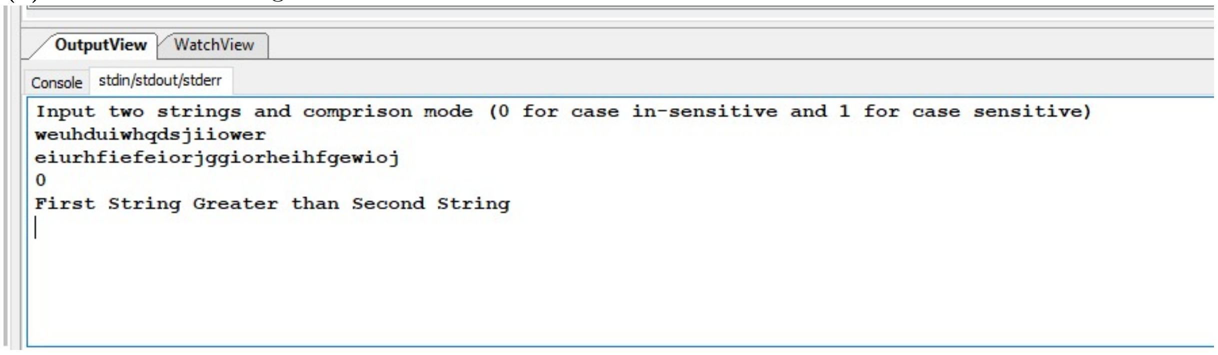
(v) This label is used to take input from the console by reading each character until we reach the new line character. Also it adds a null terminating character after the string reading is over that helps in comparison of the strings.

```
read_input:
    mov r0,#0x06
    ldr r1,=read_char
    swi 0x123456
    ldr r0,[r1,#4]
    ldrb r3,[r0]
    strb r3,[r2],#1
    cmp r3,#0x0d
    bne read_input
    mov r3,#0x00
    strb r3,[r2],#1
    mov pc,lr
```

3 Test Inputs and Results

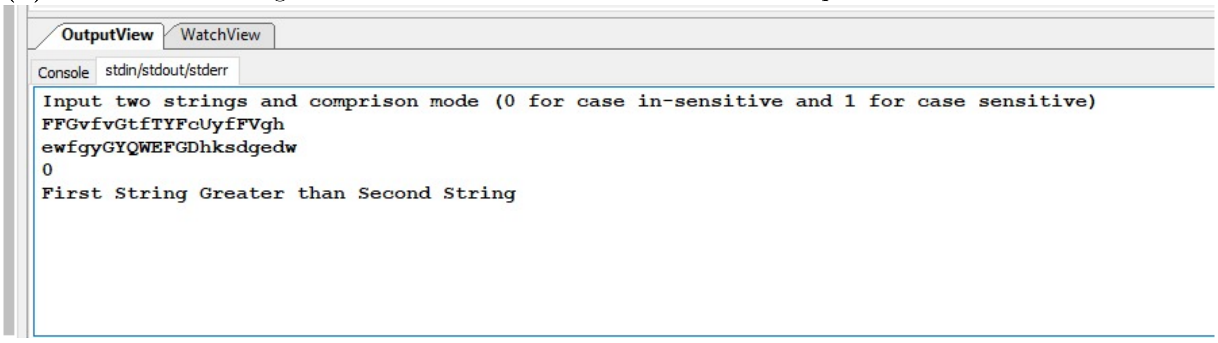
The above program was tested against some random inputs and following results were observed:

(a) Two Random strings:

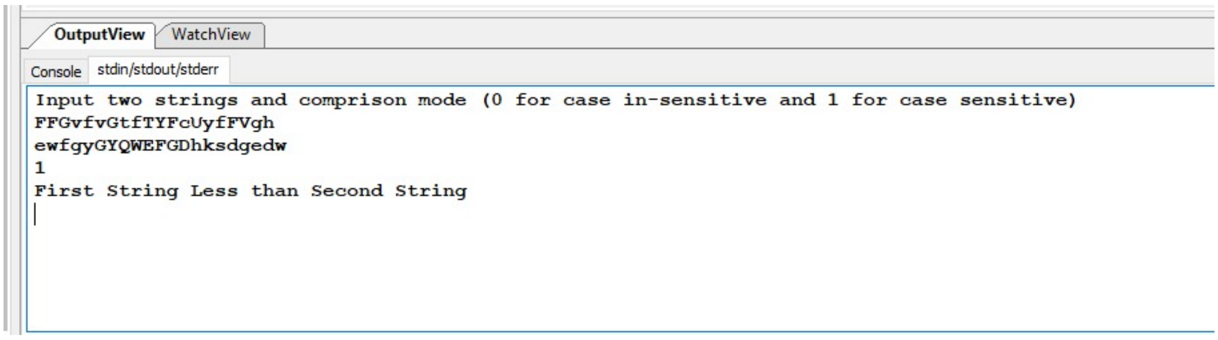


```
Input two strings and comprison mode (0 for case in-sensitive and 1 for case sensitive)
weuhduiwhqdsjiiower
eiurhfiefeiorjggiorheihfgewioj
0
First String Greater than Second String
```

(b) Two Random strings with case sensitive and case in-sensitive comparison:

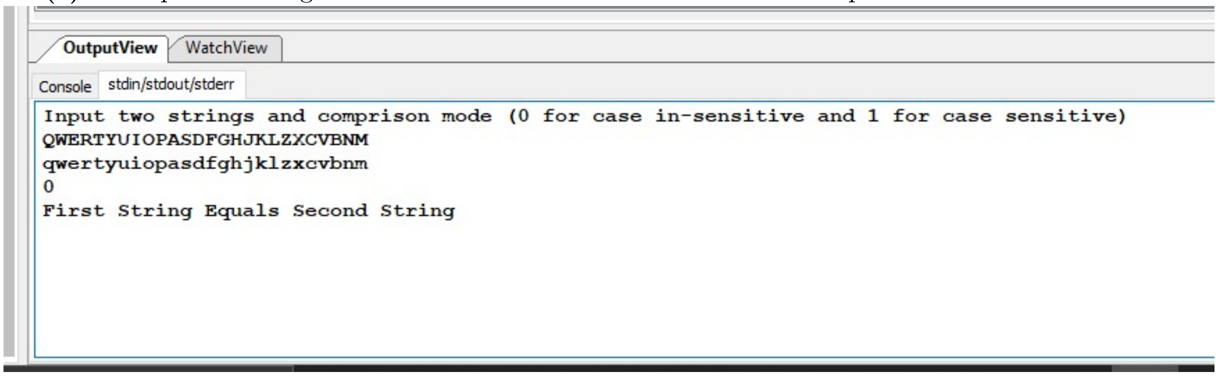


```
Input two strings and comprison mode (0 for case in-sensitive and 1 for case sensitive)
FFGvfvGtfTYFcUyfFVgh
ewfgyGYQWEFGDhksdgedw
0
First String Greater than Second String
```

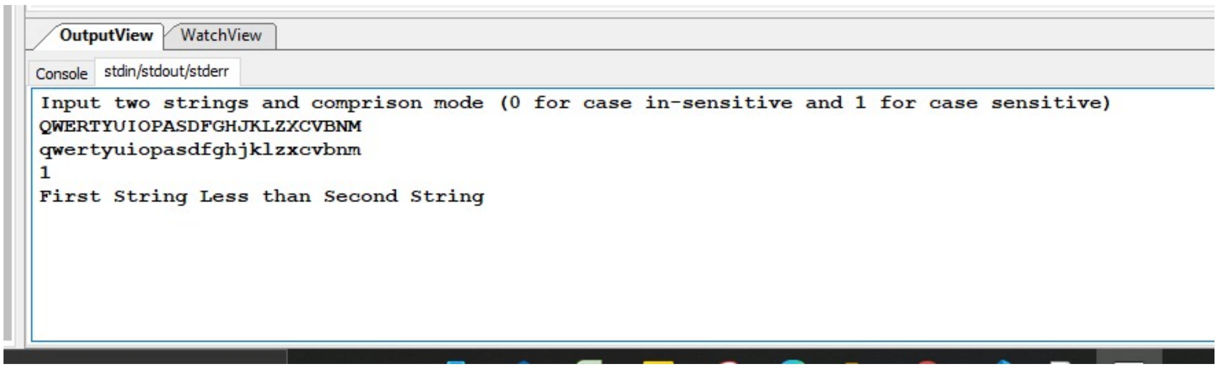


```
Input two strings and comprison mode (0 for case in-sensitive and 1 for case sensitive)
FFGvfvGtfTYFcUyfFVgh
ewfgyGYQWEFGDhksdgedw
1
First String Less than Second String
```

(c) Two specific Strings with case sensitive and case in-sensitive comparison:

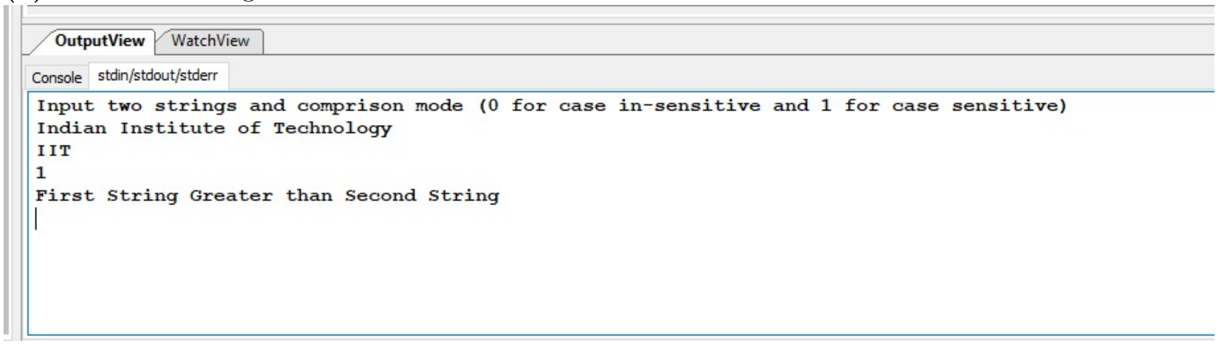


```
Input two strings and comprison mode (0 for case in-sensitive and 1 for case sensitive)
QWERTYUIOPASDFGHJKLZXCVBNM
qwertyuiopasdfghjklzxcvbnm
0
First String Equals Second String
```

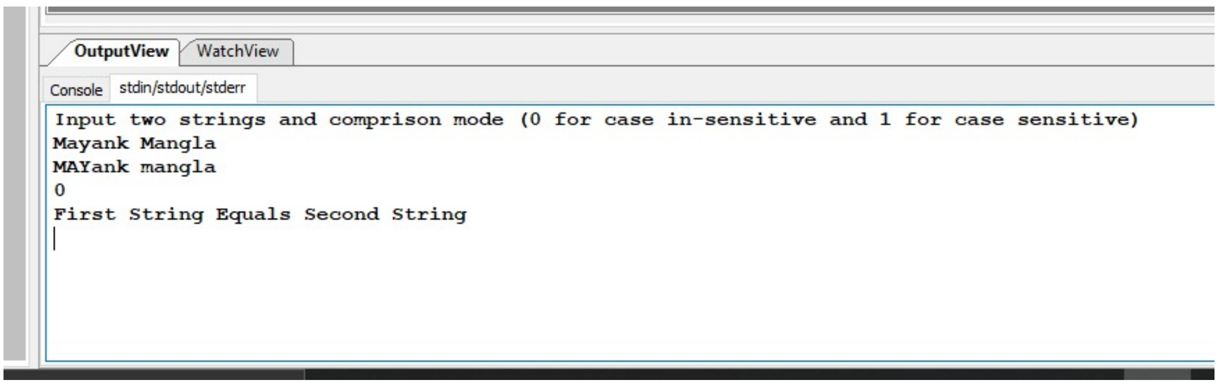


```
Input two strings and comprison mode (0 for case in-sensitive and 1 for case sensitive)
QWERTYUIOPASDFGHJKLZXCVBNM
qwertyuiopasdfghjklzxcvbnm
1
First String Less than Second String
```

(d) Two name Strings:



```
Input two strings and comprison mode (0 for case in-sensitive and 1 for case sensitive)
Indian Institute of Technology
IIT
1
First String Greater than Second String
```



```
Input two strings and comprison mode (0 for case in-sensitive and 1 for case sensitive)
Mayank Mangla
MAYank mangla
0
First String Equals Second String
```

From the above results it can be concluded that the program gives the correct output for any input. That means, when we input any two strings and the comparison made then we can get the result that which of the string is greater in lexicographic order(or dictionary order) and it will be written after the smaller string in dictionary order.