COL-216

Divyansh Mittal , 2020CS10342

Assignment 1

Computer Architecture

2022-01-27

cs1200342@iitd.ac.in

1. Assignment 1

a) Stage 3

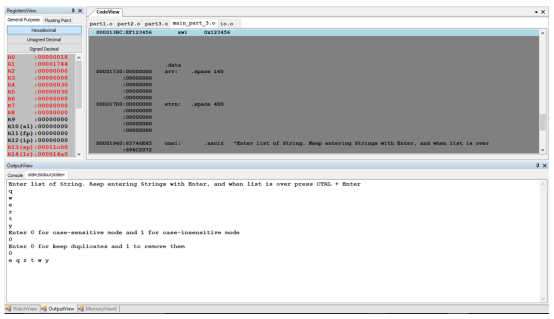
- **Input:** To call the sort function, r_0 is start of list 1, r_1 has size of it ,r_2 can be used to specify comparison mode, and r_3 for removal of duplicate.

  When running directly , run main_ part2.s Enter a list by separating strings by enter. When done with list 1, press Ctrl + Enter [1]. Then enter 0 or 1 for case sensitivity mode and similarly for removal or not of duplicates.

- **Output:** The r_0, r_1 has the locn and size of sorted list.

  If directly running you will get the sorted list printed. [2]

Working: Instead of sizes, dealing with pointers is easy. So in my merge_sort function I call a helper function sort that actually implements the algorithm. It takes in r_0 as start of list 1, r_3 as end of list 1, r_4 and r_5 for case mode and duplicate removal. It return r_0 as start of sorted list and r_1 points to its end.
In the function, I find the middle pointer by doing r_3-r_0, and right shift by 3 , and left shift by 2^3. r_1 stores this value and r_2 stores r_1+ 4. Now I call sort on r_0,r_1 section and r_2,r_3 section.[4]
If $r_0 = r_3$, I simply return back.
The sorted list pointers are then moved to r_0,r_1 and r_2,r_3 respectively. Now merge is called on these. Which finally return r_0,r_1 as pointer to the final sorted list. This is then returned, and the r_1, which has the end list pointer, is converted to store the size by doing r_1-r_0 and right shift by 2. [5]



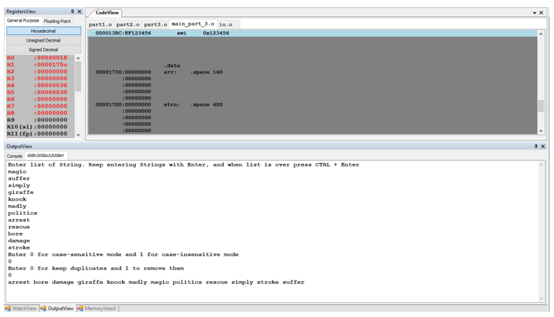
```
R0       :00000018
R1       :00001744
R2       :00000000
R3       :00000000
R4       :00000030
R5       :00000030
R6       :00000000
R7       :00000000
R8       :00000000
R9       :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00011c00
R14(lr):000014a0
```

```
000013BC:EF123456        swi     0x123456

                    .data
00001730:00000000   arr:    .space 160
        :00000000
        :00000000
        :00000000
        :00000000
000017D0:00000000   strn:   .space 400
        :00000000
        :00000000
        :00000000
        :00000000

00001960:65746E45   onei:       .asciz   "Enter list of String. Keep entering Strings with Enter, and when list is over
        :696C2072
```

```
Enter list of String. Keep entering Strings with Enter, and when list is over press CTRL + Enter
q
w
e
r
t
y
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
0
Enter 0 for keep duplicates and 1 to remove them
0
e q r t w y
```



```
R0       :00000018
R1       :0000175c
R2       :00000000
R3       :00000000
R4       :00000030
R5       :00000030
R6       :00000000
R7       :00000000
R8       :00000000
R9       :00000000
R10(sl):00000000
R11(fp):00000000
```

```
000013BC:EF123456        swi     0x123456

                    .data
00001730:00000000   arr:    .space 160
        :00000000
        :00000000
        :00000000
        :00000000
000017D0:00000000   strn:   .space 400
        :00000000
        :00000000
        :00000000
        :00000000
```

```
Enter list of String. Keep entering Strings with Enter, and when list is over press CTRL + Enter
magic
suffer
simply
giraffe
knock
madly
politics
arrest
rescue
bore
damage
stroke
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
0
Enter 0 for keep duplicates and 1 to remove them
0
arrest bore damage giraffe knock madly magic politics rescue simply stroke suffer
```

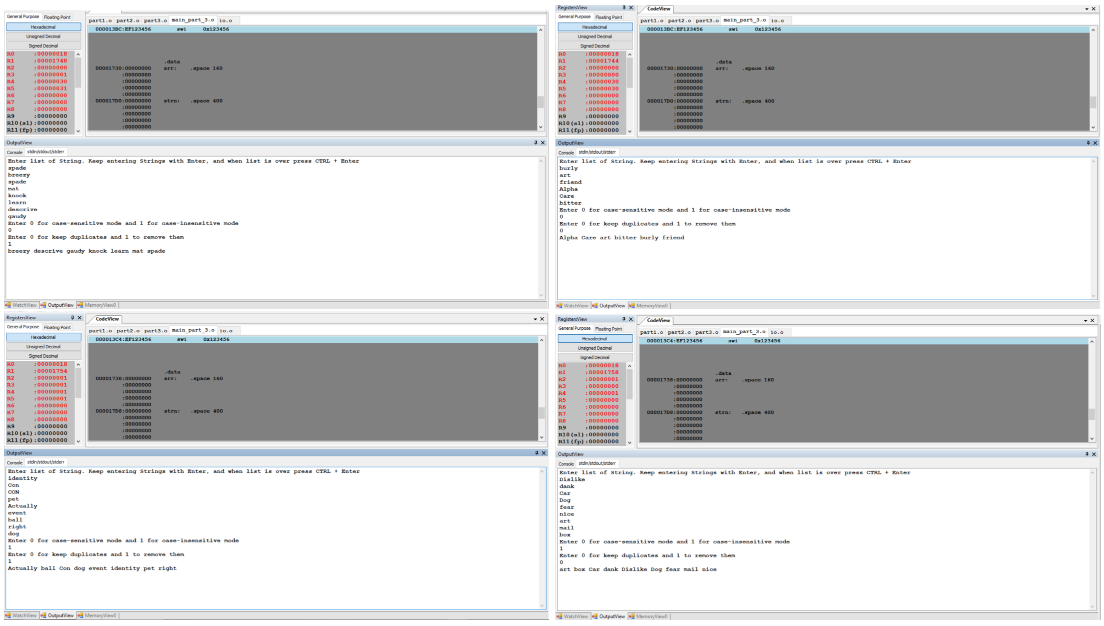
[1] For windows, pressing enter sends '\ r' char and pressing ctrl + enter registers as '\ n'. The io replaces '\ r','\ n' with the null character

[2] The IO prints space after the null char

[3] So that I get aligned pointer location

[4] This is the reason working with pointers is easier, now I don't need to convert to size of subsection and call sort

[5] As each word takes 4 address, so right shift divides by 4



```
R0      :00000018
R1      :00001748
R2      :00000000
R3      :00000001
R4      :00000030
R5      :00000031
R6      :00000000
R7      :00000000
R8      :00000000
R9      :00000000
R10(sl):00000000
R11(fp):00000000
```

```
000013BC:EF123456        swi      0x123456

                    .data
00001730:00000000   arr:     .space 160
        :00000000
        :00000000
        :00000000
        :00000000
000017D0:00000000   strn:    .space 400
        :00000000
        :00000000
        :00000000
        :00000000
```

```
Enter list of String. Keep entering Strings with Enter, and when list is over press CTRL + Enter
spade
breezy
spade
mat
knock
learn
descrive
gaudy
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
0
Enter 0 for keep duplicates and 1 to remove them
1
breezy descrive gaudy knock learn mat spade
```

```
R0      :00000018
R1      :00001744
R2      :00000000
R3      :00000000
R4      :00000030
R5      :00000030
R6      :00000000
R7      :00000000
R8      :00000000
R9      :00000000
R10(sl):00000000
R11(fp):00000000
```

```
000013BC:EF123456        swi      0x123456

                    .data
00001730:00000000   arr:     .space 160
        :00000000
        :00000000
        :00000000
        :00000000
000017D0:00000000   strn:    .space 400
        :00000000
        :00000000
        :00000000
        :00000000
```

```
Enter list of String. Keep entering Strings with Enter, and when list is over press CTRL + Enter
burly
art
friend
Alpha
Care
bitter
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
0
Enter 0 for keep duplicates and 1 to remove them
0
Alpha Care art bitter burly friend
```

```
R0      :00000018
R1      :00001754
R2      :00000001
R3      :00000001
R4      :00000001
R5      :00000001
R6      :00000000
R7      :00000000
R8      :00000000
R9      :00000000
R10(sl):00000000
R11(fp):00000000
```

```
000013C4:EF123456        swi      0x123456

                    .data
00001738:00000000   arr:     .space 160
        :00000000
        :00000000
        :00000000
        :00000000
000017D8:00000000   strn:    .space 400
        :00000000
        :00000000
        :00000000
        :00000000
```

```
Enter list of String. Keep entering Strings with Enter, and when list is over press CTRL + Enter
identity
Con
CON
pet
Actually
event
ball
right
dog
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
1
Enter 0 for keep duplicates and 1 to remove them
1
Actually ball Con dog event identity pet right
```

```
R0      :00000018
R1      :00001758
R2      :00000001
R3      :00000000
R4      :00000001
R5      :00000000
R6      :00000000
R7      :00000000
R8      :00000000
R9      :00000000
R10(sl):00000000
R11(fp):00000000
```

```
000013C4:EF123456        swi      0x123456

                    .data
00001738:00000000   arr:     .space 160
        :00000000
        :00000000
        :00000000
        :00000000
000017D8:00000000   strn:    .space 400
        :00000000
        :00000000
        :00000000
        :00000000
```

```
Enter list of String. Keep entering Strings with Enter, and when list is over press CTRL + Enter
Dislike
dank
Car
Dog
fear
nice
art
mail
box
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
1
Enter 0 for keep duplicates and 1 to remove them
0
art box Car dank Dislike Dog fear mail nice
```

Figure 1: Various test cases

b) Stage 2

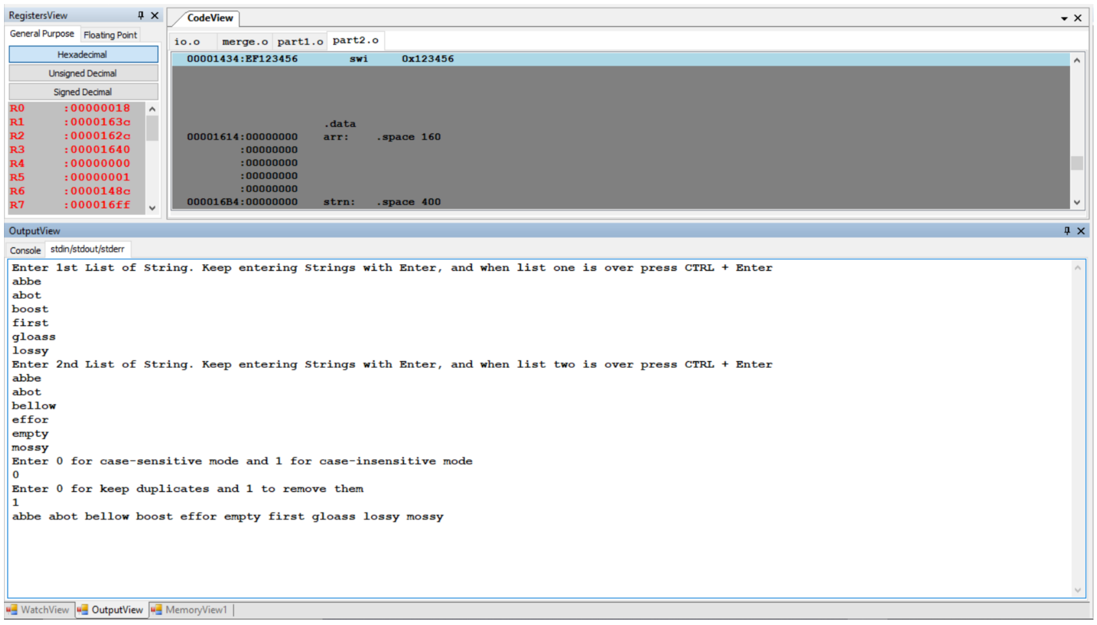
- **Input:** To call the merge function, r_0 is start of list 1, r_1 has size of it r_2 has pointer to list 2, r_3 has size of it, r_4 can be used to specify comparison mode, and r_5 for removal of duplicate.

  When running directly , run main_ part2.s Enter a list by separating strings by enter. When done with list 1, press Ctrl + Enter [6]. Do the same for list 2. Then enter 0 or 1 for case sensitivity mode and similarly for removal or not of duplicates.

- **Output:** The r_0, r_1 has the locn and size of merged list. The merge uses temporary space to sort, but replaces the original list too, hence can be easily used for merge sort

  If directly running you will get the sorted list printed. [7]

Working: It works via a two pointer method. In the function merge, initially r_0 points to list 1 start, r_1 points to list 1 end, r_2 points at list 2 start and r_3 points to list 2 end. String at r_0,r_2 pointer are compared. Whatever is smaller, r_6 places pointer at a temp locn , advances and the one which is placed is also advanced. This loop continues till r_0 becomes more than r_1 or r_2 becomes more than r_3. In this case the rest of pointers are copied at the temp locn. If we have to remove duplicate, and a duplicate occours, the val at r_0 is placed at r_6 and both r_0,r_2 advance together. The final merge list is coppied back at the original r_0 that was passed and r_1 holds size of sorted list.



```
R0    :00000018
R1    :0000163c
R2    :0000162c
R3    :00001640
R4    :00000000
R5    :00000001
R6    :0000148c
R7    :000016ff
```

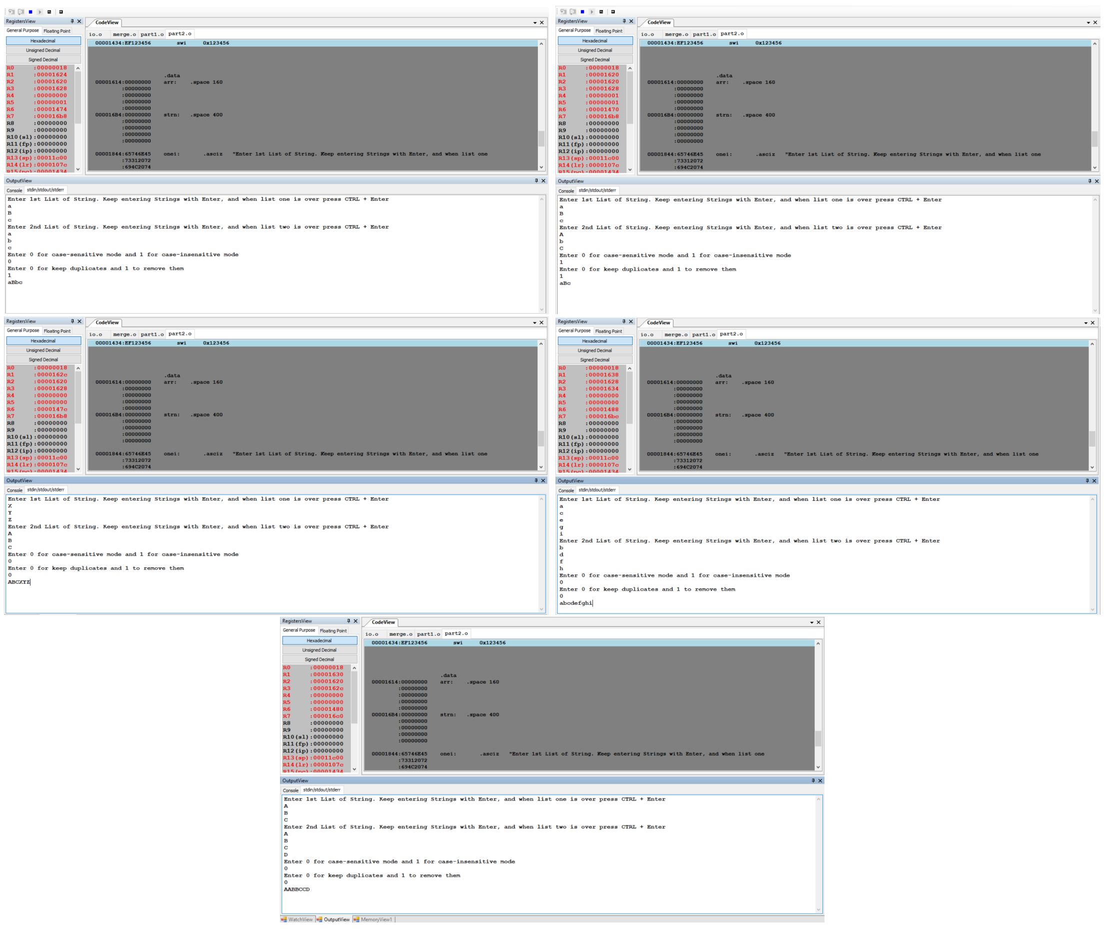
```
 00001434:EF123456        swi     0x123456


                        .data
 00001614:00000000      arr:    .space 160
          :00000000
          :00000000
          :00000000
          :00000000
 000016B4:00000000      strn:   .space 400
```

```
Enter 1st List of String. Keep entering Strings with Enter, and when list one is over press CTRL + Enter
abbe
abot
boost
first
gloass
lossy
Enter 2nd List of String. Keep entering Strings with Enter, and when list two is over press CTRL + Enter
abbe
abot
bellow
effor
empty
mossy
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
0
Enter 0 for keep duplicates and 1 to remove them
1
abbe abot bellow boost effor empty first gloass lossy mossy
```

[6]For windows, pressing enter sends '\ r' char and pressing ctrl + enter registers as '\ n'. The io replaces '\ r','\ n' with the null character

[7]The IO does not add any "\ n" or space char. So enter space with the word to get spaced out output. In the memory the strings stored are null terminated



```
RegistersView
General Purpose  Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0      :00000018
R1      :00001624
R2      :00001620
R3      :00001628
R4      :00000000
R5      :00000001
R6      :00001474
R7      :000016b8
R8      :00000000
R9      :00000000
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00011c00
R14(lr) :0000107c
R15(pc) :00001434
```

```
CodeView
io.o  merge.o  part1.o  part2.o
 00001434:EF123456        swi      0x123456

                        .data
 00001614:00000000    arr:    .space 160
         :00000000
         :00000000
         :00000000
         :00000000
 000016B4:00000000    strn:   .space 400
         :00000000
         :00000000
         :00000000
         :00000000

 00001844:65746E45    onei:       .asciz   "Enter 1st List of String. Keep entering Strings with Enter, and when list one
         :73312072
         :694C2074
```

```
OutputView
Console  stdin/stdout/stderr
Enter 1st List of String. Keep entering Strings with Enter, and when list one is over press CTRL + Enter
a
B
c
Enter 2nd List of String. Keep entering Strings with Enter, and when list two is over press CTRL + Enter
a
b
c
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
0
Enter 0 for keep duplicates and 1 to remove them
1
aBbc
```

```
RegistersView
General Purpose  Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0      :00000018
R1      :00001620
R2      :00001620
R3      :00001628
R4      :00000001
R5      :00000001
R6      :00001470
R7      :000016b8
R8      :00000000
R9      :00000000
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00011c00
R14(lr) :0000107c
R15(pc) :00001434
```

```
CodeView
io.o  merge.o  part1.o  part2.o
 00001434:EF123456        swi      0x123456

                        .data
 00001614:00000000    arr:    .space 160
         :00000000
         :00000000
         :00000000
         :00000000
 000016B4:00000000    strn:   .space 400
         :00000000
         :00000000
         :00000000
         :00000000

 00001844:65746E45    onei:       .asciz   "Enter 1st List of String. Keep entering Strings with Enter, and when list one
         :73312072
         :694C2074
```

```
OutputView
Console  stdin/stdout/stderr
Enter 1st List of String. Keep entering Strings with Enter, and when list one is over press CTRL + Enter
a
B
c
Enter 2nd List of String. Keep entering Strings with Enter, and when list two is over press CTRL + Enter
A
b
C
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
1
Enter 0 for keep duplicates and 1 to remove them
1
aBc
```

```
RegistersView
General Purpose  Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0      :00000018
R1      :0000162c
R2      :00001620
R3      :00001628
R4      :00000000
R5      :00000000
R6      :0000147c
R7      :000016b8
R8      :00000000
R9      :00000000
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00011c00
R14(lr) :0000107c
R15(pc) :00001434
```

```
CodeView
io.o  merge.o  part1.o  part2.o
 00001434:EF123456        swi      0x123456

                        .data
 00001614:00000000    arr:    .space 160
         :00000000
         :00000000
         :00000000
         :00000000
 000016B4:00000000    strn:   .space 400
         :00000000
         :00000000
         :00000000
         :00000000

 00001844:65746E45    onei:       .asciz   "Enter 1st List of String. Keep entering Strings with Enter, and when list one
         :73312072
         :694C2074
```

```
OutputView
Console  stdin/stdout/stderr
Enter 1st List of String. Keep entering Strings with Enter, and when list one is over press CTRL + Enter
X
Y
Z
Enter 2nd List of String. Keep entering Strings with Enter, and when list two is over press CTRL + Enter
A
B
C
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
0
Enter 0 for keep duplicates and 1 to remove them
0
ABCXYZ
```

```
RegistersView
General Purpose  Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0      :00000018
R1      :00001638
R2      :00001628
R3      :00001634
R4      :00000000
R5      :00000000
R6      :00001488
R7      :000016bc
R8      :00000000
R9      :00000000
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00011c00
R14(lr) :0000107c
R15(pc) :00001434
```

```
CodeView
io.o  merge.o  part1.o  part2.o
 00001434:EF123456        swi      0x123456

                        .data
 00001614:00000000    arr:    .space 160
         :00000000
         :00000000
         :00000000
         :00000000
 000016B4:00000000    strn:   .space 400
         :00000000
         :00000000
         :00000000
         :00000000

 00001844:65746E45    onei:       .asciz   "Enter 1st List of String. Keep entering Strings with Enter, and when list one
         :73312072
         :694C2074
```

```
OutputView
Console  stdin/stdout/stderr
Enter 1st List of String. Keep entering Strings with Enter, and when list one is over press CTRL + Enter
a
c
e
g
i
Enter 2nd List of String. Keep entering Strings with Enter, and when list two is over press CTRL + Enter
b
d
f
h
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
0
Enter 0 for keep duplicates and 1 to remove them
0
abcdefghi
```

```
RegistersView
General Purpose  Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0      :00000018
R1      :00001630
R2      :00001620
R3      :0000162c
R4      :00000000
R5      :00000000
R6      :00001480
R7      :000016c0
R8      :00000000
R9      :00000000
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00011c00
R14(lr) :0000107c
R15(pc) :00001434
```

```
CodeView
io.o  merge.o  part1.o  part2.o
 00001434:EF123456        swi      0x123456

                        .data
 00001614:00000000    arr:    .space 160
         :00000000
         :00000000
         :00000000
         :00000000
 000016B4:00000000    strn:   .space 400
         :00000000
         :00000000
         :00000000
         :00000000

 00001844:65746E45    onei:       .asciz   "Enter 1st List of String. Keep entering Strings with Enter, and when list one
         :73312072
         :694C2074
```

```
OutputView
Console  stdin/stdout/stderr
Enter 1st List of String. Keep entering Strings with Enter, and when list one is over press CTRL + Enter
A
B
C
Enter 2nd List of String. Keep entering Strings with Enter, and when list two is over press CTRL + Enter
A
B
C
D
Enter 0 for case-sensitive mode and 1 for case-insensitive mode
0
Enter 0 for keep duplicates and 1 to remove them
0
AABBCCD
```

Figure 2: Various test cases

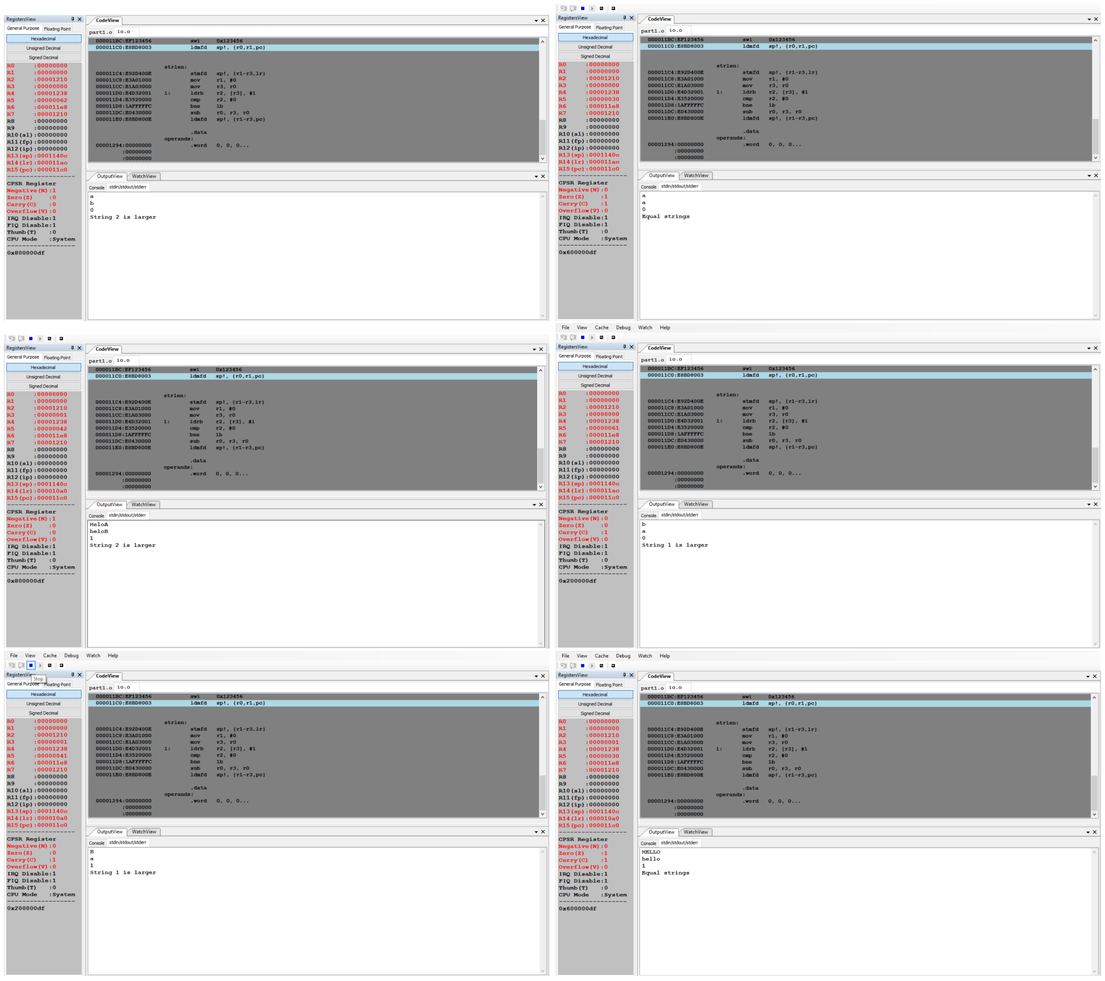
c) Stage 1

- **Input:** To call the comparison function, r_1 and r_2 can be used to pass pointer to string 1 and string 2. r_3 can be used to specify comparison mode. Place 0 for case-sensitive mode and 1 for case-insensitive mode

  If directly running part1.s, enter string 1, string 2, and comparison mode (0 or 1) via stdin[8]

- **Output:** If you call the comparison function, result is stored in r_0. -1 is returned if string2 > string 1, 0 if string1 = string2 and 1 if string1 < string 2

  If directly running part1.s, stdout text to tell which string is bigger



```
000011BC:EF123456          swi     0x123456
000011C0:E8BD8003          ldmfd   sp!, {r0,r1,pc}

                  strlen:
000011C4:E92D400E          stmfd   sp!, {r1-r3,lr}
000011C8:E3A01000          mov     r1, #0
000011CC:E1A03000          mov     r3, r0
000011D0:E4D32001   1:     ldrb    r2, [r3], #1
000011D4:E3520000          cmp     r2, #0
000011D8:1AFFFFFC          bne     1b
000011DC:E0430000          sub     r0, r3, r0
000011E0:E8BD800E          ldmfd   sp!, {r1-r3,pc}

                           .data
                  operands:
00001294:00000000          .word   0, 0, 0...
        :00000000
        :00000000
```

```
a
b
0
String 2 is larger
```

```
a
a
0
Equal strings
```

```
HeloA
heloB
1
String 2 is larger
```

```
b
a
0
String 1 is larger
```

```
B
a
1
String 1 is larger
```

```
HELLO
hello
1
Equal strings
```

Figure 3: Various test cases

[8]In direct run r_0 will get 0 as print is called