BIRZEIT UNIVERSITY

Computer Organization

ENCS2380

Assembly Project Report

By : Baraa Khanfar

ID : 1210640

Section : 2

Task 1 : Creating the strings :

I used an read-only data area for strings , and the code is :

```
  AREA MYDATA, DATA, READONLY
STR1 DCB "BBaRaa RBBrRM",0
STR2 DCB "BAhmaRd MohammradBB",0
```

And I used read and write data area for the converted ,encrypted string , counters of converted letters and the common characters counter and the code is :

```
 AREA WRITEDATA,DATA,READWRITE
;you can change the values of space 30 if you want to do it in a larger string
TEXT1 space 30        ;the samll letter of STR1
Count1 DCB 0          ;count of converted letters from STR1
TEXT2 space 30        ;the samll letter of STR2
Count2 DCB 0          ;count of converted letters from STR2
COMMON DCB 0          ;count of common charcters between them
ENCRYPT1 space 30
ENCRYPT2 space 30
```

Task 2 : Converting the strings to lower case :

1. In the main I loaded the registers with the starting address value of the first string and the converted string ,then I called the procedure , like this :

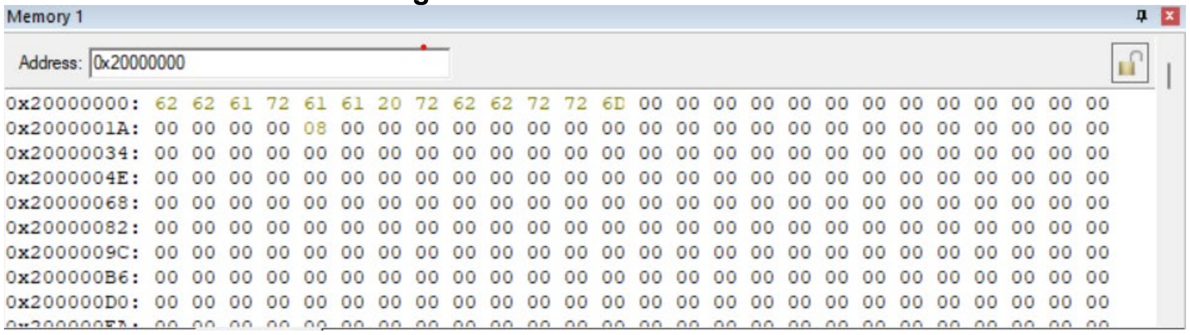
```
  MOV R3, #0 ; initialize counter for converted letters ..
  LDR R0, =STR1 ; R4 points to the first input string
  LDR R1, =TEXT1; R5 points to the first output string
  LDR R4,=Count1;R2 points to the first string converted charcters counter
  BL ToLower
```

   And the same steps for the second string

2. The code of the procedure is :

```
ToLower PROC
convert_loop
 LDRB R2, [R0], #1 ; load a single character from the input string
 CMP R2, #0 ; compare the character with 0
 BEQ lowercase_end ;  the conversion is done
 CMP R2, #'A' ; compare the character with 'A'
 BLO store_char ;if its lower it will store it
 CMP R2,#'Z' ; if its higher it will store it
 BHI store_char
 ADD R2, R2,#32 ;else it will add 32 to it to convert to lower case
 ADD R3, R3,#1;counter of converted letters
store_char
 STRB R2, [R1], #1 ; store the converted character in the output string
 B convert_loop ; go to the next character
lowercase_end
 STRB R3,[R4] ; store the number of converted letters in the location pointed to by
R2
 BX LR
 ENDP
```

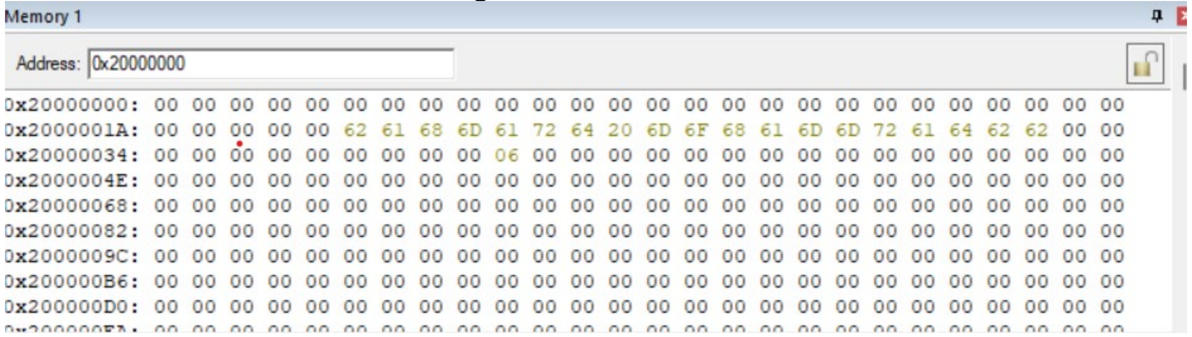
3. The result of the first string is :



```
Memory 1
Address: 0x20000000
0x20000000: 62 62 61 72 61 61 20 72 62 62 72 72 6D 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000001A: 00 00 00 00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000034: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000004E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000068: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000082: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000009C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000B6: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

The result is : 62 62 61 72 61 61 20 72 62 62 72 72 6D which represents :
bbaraa rbbrrm which is the lower case version of : BBaRaa RBBrRM
and 8 is the count of converted charcters

4. The result for the second string is :



```
Memory 1
Address: 0x20000000
0x20000000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000001A: 00 00 00 00 00 62 61 68 6D 61 72 64 20 6D 6F 68 61 6D 6D 72 61 64 62 62 00 00
0x20000034: 00 00 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000004E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000068: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000082: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000009C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000B6: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

The result is : 62 61 68 6D 61 72 64 20 6D 6F 68 61 6D 6D 72 61 64 62 62
which represents :
bahmard mohammradbb which is the lower case version of : BAhmaRd MohammradBB
and 6 is the count of converted charcters

Task 3 : Counting common characters between the strings :

1. In the main I loaded the registers with the starting address value of the converted version of the first string and the counter address, like this :

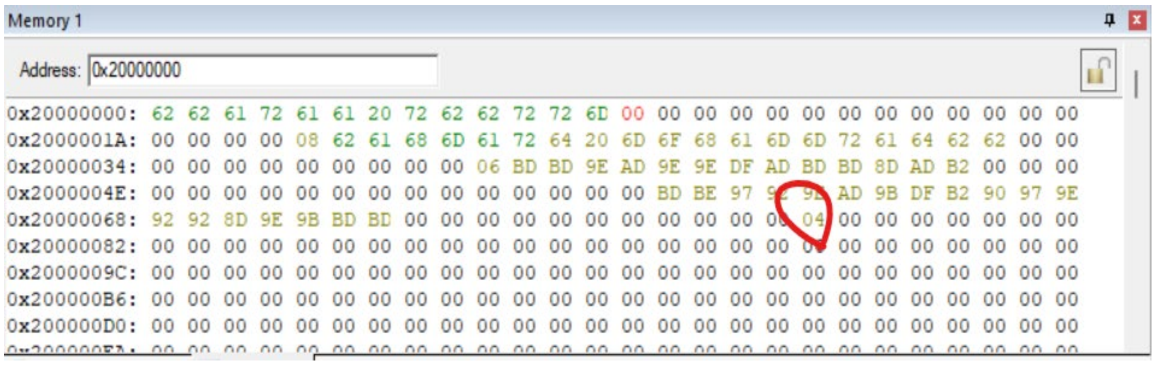
```
MOV R5,#0          ; the counter of equal characters in str2 from 1
MOV R11,#0         ; counter for the repeated characters in the first string because it
 will counted more than one time
LDR R0,=TEXT1          ;R0 points to the converted version of STR1
LDR R2,=COMMON     ;0 R2 ponits to the common counter
BL Common_Count
```

If the string isn't already converted you can just call the tolower procedure and then count them , it's a lot easier than checking every character

2. **The code of the procedure is :**

```
Common_Count PROC
common_count_loop1
 LDRB R3,[R0],#1          ;R3 contains the first character in the first string
 CMP R3,#0                ;check if the end of the string is reached
 BEQ store_common         ;if yes it will store the count
 LDR R1,=TEXT2            ; intialize the R1 in each loop for iterating through it
common_count_loop2
 LDRB R4,[R1],#1          ; load a single character from the second string
 CMP R4, #0               ; check if the end of the string has been reached
 BEQ common_count_loop1  ;if yes,go to the next character in the first string
 CMP R3,#' '              ;not counting the spaces
 BEQ common_count_loop1
 CMP R3,R4
 BEQ addition            ;if they are equal increment the counter and check how many
times the character in R4 exists in the first string
 B common_count_loop2
addition
 ADD R5,R5,#1            ;increment thhe counter of the common charcters between
;them each repeated character in the first
 ;string will be counted more than one time
 MOV R6,R0
count_same_characters_loop
 LDRB R7, [R6], #1  ; load a single character from the location that R6 is currently
pointing to
 CMP R7, #0           ; check if the end of the string has been reached
 BEQ common_count_loop1  ;if yes , return to the first loop
 CMP R7,R4
 BEQ add_count       ;increment the counter of reapated charcters
 B count_same_characters_loop    ;go to the next character
add_count
 ADD R11,R11, #1 ; add to the count
 B common_count_loop1   ;return to the first loop
store_common
 SUB R5,R5,R11; ;subtract the count of repated charcters from R5
 STRB R5,[R2] ;store the count
 BX LR
 ENDP
```

3. **The result will be 4 , the first string is** BBaRaa RBBrRM and the second is BAhmaRd MohammradBB **and the common characters between them is 'A','B','R' and 'M'**



Memory 1

Address: 0x20000000

```
0x20000000: 62 62 61 72 61 61 20 72 62 62 72 72 6D 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000001A: 00 00 00 00 08 62 61 68 6D 61 72 64 20 6D 6F 68 61 6D 6D 72 61 64 62 62 00 00
0x20000034: 00 00 00 00 00 00 00 00 00 06 BD BD 9E AD 9E 9E DF AD BD BD 8D AD B2 00 00 00
0x2000004E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 BD BE 97 9[illegible] 9[illegible] AD 9B DF B2 90 97 9E
0x20000068: 92 92 8D 9E 9B BD BD 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00
0x20000082: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000009C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000B6: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Task 4 : Encrypting the strings :

1. In the main I loaded the registers with the starting address value of the first string and the encrypted string location, like this :

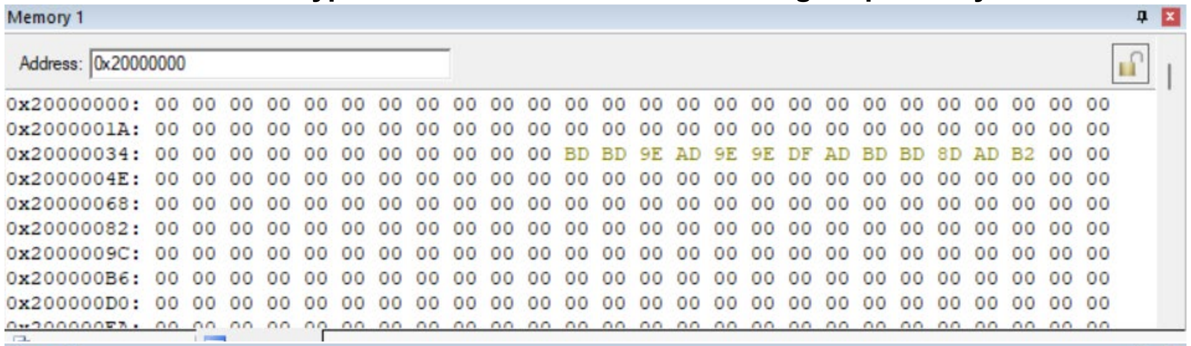
```
   LDR R0, =STR1 ; R0 points to the input string
   LDR R1, =ENCRYPT1 ; R1 points to the output string
   BL Encrypt
```

And the same steps for the second string

2. The code of the procedure is :

```
Encrypt PROC
encrypt_loop
 LDRB R2, [R0], #1     ; load a single character from the input string
 CMP R2, #0            ; check if the end of the string has been reached
 BEQ END_               ; if 0,end
 MVN R2, R2             ; invert the bits of the character in R2 and store in R2
 STRB R2, [R1], #1      ; store the encrypted character in the output string
 B encrypt_loop
END_
  BX LR
  ENDP
```

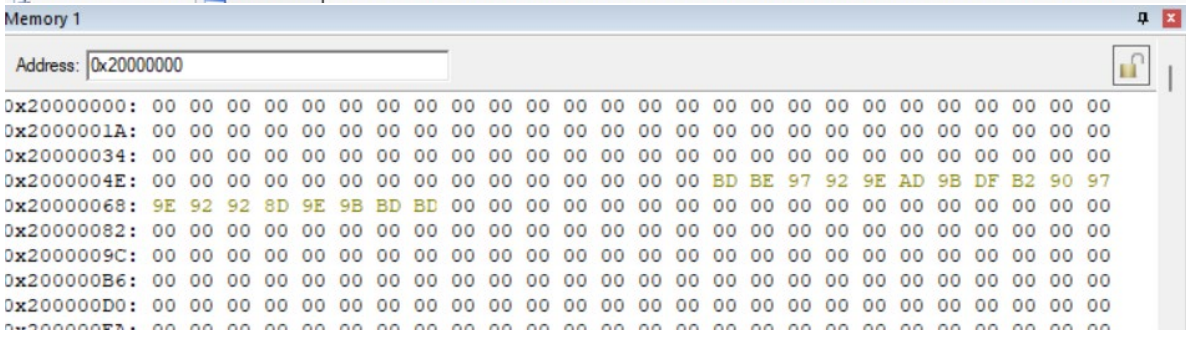
3. The result of the encryption of the first and second string respectively :



Memory 1

Address: 0x20000000

```
0x20000000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000001A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000034: 00 00 00 00 00 00 00 00 00 00 00 BD BD 9E AD 9E 9E DF AD BD BD 8D AD B2 00 00
0x2000004E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000068: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000082: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000009C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000B6: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



Memory 1

Address: 0x20000000

```
0x20000000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000001A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000034: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000004E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 BD BE 97 92 9E AD 9B DF B2 90 97
0x20000068: 9E 92 92 8D 9E 9B BD BD 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000082: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000009C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000B6: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```