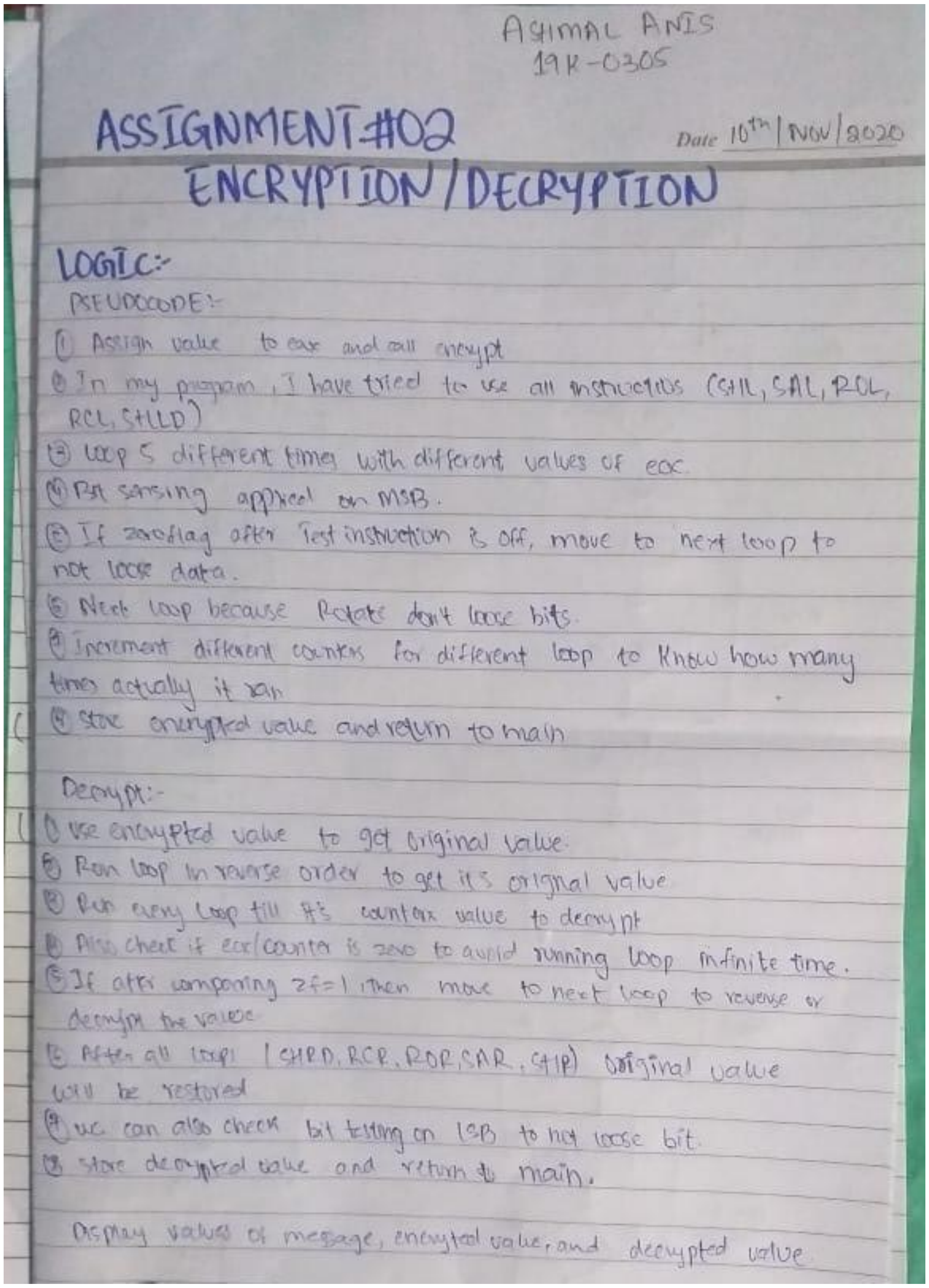
Ashmal Anis 19K-0305 Sec: H

COAL ASSIGNMENT # 02

TASK: Encryption/ Decryption

LOGIC:



ASHMAL ANIS
19K-0305

ASSIGNMENT#02 Date 10th/NOV/2020
ENCRYPTION/DECRYPTION

LOGIC:-
PSEUDOCODE:-

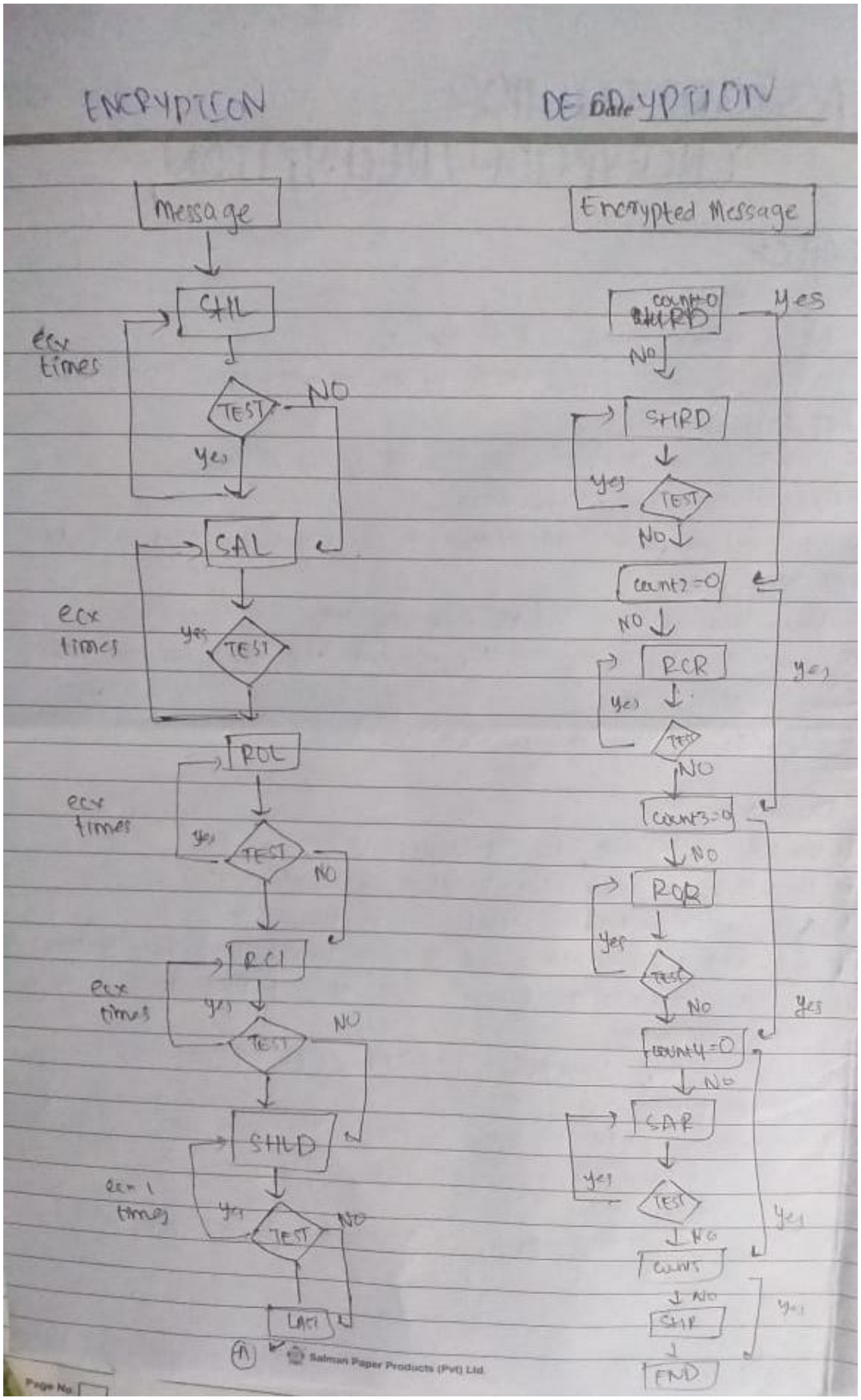
```
1 Assign value to eax and call encrypt
2 In my program, I have tried to use all instructions (SHL, SAL, ROL, RCL, SHLD)
3 Loop 5 different times with different values of eax.
4 Bit sensing applied on MSB.
5 If zeroflag after Test instruction is off, move to next loop to not loose data.
6 Next loop because Rotate don't loose bits.
7 Increment different counters for different loop to know how many times actually it ran
8 Store encrypted value and return to main
```

Decrypt:-

```
1 Use encrypted value to get original value.
2 Run loop in reverse order to get it's orignal value
3 Run every Loop till it's counter value to decrypt
4 Also check if ecx/counter is zero to avoid running loop infinite time.
5 If after comparing zf=1 then move to next loop to reverse or decrypt the value
6 After all loops (SHRD, RCR, ROR, SAR, SHR) original value will be restored
7 We can also check bit testing on LSB to not loose bit.
8 Store decrypted value and return to main.
```

Display values of message, encrypted value, and decrypted value

Ashmal Anis 19K-0305 Sec: H



CODE:

```
TITLE Assignment2 (test.asm)
INCLUDE IRVINE32.INC

.data
message dword 0581D723h
encrypted dword ?
decrypted dword ?

count1 byte 0
count2 byte 0
count3 byte 0
count4 byte 0
count5 byte 0

m1 byte "Original Message Value : ",0
m2 byte "Value after encryption : ",0
m3 byte "Value after decryption : ",0

.code
main proc

; displaying original value
mov edx, offset m1
call writeString
mov eax, message
call WriteHex
call crlf

call encrypt                    ; calling function
mov edx, offset m2              ; displaying encrypted value
call writeString
mov eax, encrypted
call WriteHex
call crlf

call decrypt                    ; calling decrypt function
mov edx, offset m3              ; displaying decrypted value
call writeString
mov eax, decrypted
```

```
call WriteHex
call crlf

mov eax, message                    ; to display the values in register also
mov ebx, encrypted
mov ecx, decrypted
call dumpregs
exit
main endp


;a) ENCRYPTION PROCEDURE
encrypt proc

mov eax, message
mov ecx, 3
mov ebx, 0BCADE0h                   ; just a random value for applying shld

l1:                                 ; first iteration will perform shift left
        test eax, 10000000000000000000000000000000b         ; will check msb (bit sensing)
to not loose data
        jnz l2   ; if msb=1 it can move to next loop because (ROL, RCL is possible we don't loose
data in them)
        SHL eax, 1
        inc count1                                                          ; to
store the count value to accordingly decrypt, it may not run the ecx times because of bit sensing
loop l1

mov ecx, 2
l2:
        test eax, 01000000000000000000000000000000b         ; bit sensing at MSB-1 bit
also because though it will encrypt correctly but decryption becomes impossible

; for example 0101 1000 after shl = 1010 0000, but decryption = 1101 1000 (D8) data altered
        jnz l3
        test eax, 10000000000000000000000000000000b         ; bit sensing at msb also
        jnz l3
        SAL eax, 1
        inc count2
loop l2
```

```
mov ecx, 3
l3:
        ; will perform rol instruction no data is lost, it's possible
        test eax, 10000000000000000000000000000000b
        jnz l4
        ROL eax, 1
        inc count3
loop l3

mov ecx, 4
l4:
        ; will perform rotate cary left instruction no data is lost, it's possible
        test eax, 10000000000000000000000000000000b
        jnz l5
        RCL eax, 1
        inc count4
loop l4

mov ecx, 2
l5:
        ; will perfom shld instruction with ebx which was defined at the start of function
        test eax, 10000000000000000000000000000000b
        jnz last
        SHLD eax, ebx , 1
        inc count5
loop l5


last :
mov encrypted, eax
ret
encrypt endp


;b) DECYPTION PROCEDURE
decrypt proc

mov eax, encrypted
mov ebx, 0BCADE0h          ; to perform shrd
```

```
movzx ecx, count5

; it may be possible that data did not entered certain loop in instruction or haven't run ecx times,
instead countx times. If countx is 0 then loop will run infinte time thus invalid execution occurs,
we check if ecx is 0, check next loop if it was executed

cmp ecx, 0
jz j1
l5:                                 ; will check in reverse order to decrypt in systematic manner
        test eax, 00000000000000000000000000000001b
        jnz last
        SHRD eax, ebx , 1
loop l5

j1:
movzx ecx, count4
cmp ecx, 0
jz j2
l4:                                                 ; will perfrom rcr instruction and decrypt
        test eax, 00000000000000000000000000000001b
        jnz last
        RCR eax, 1
loop l4

j2:
movzx ecx, count3
cmp ecx, 0
jz j3
l3:                                                 ; will perform ror instruction
        test eax, 00000000000000000000000000000001b
        jnz last
        ROR eax, 1
loop l3


j3:
movzx ecx, count2
cmp ecx, 0
jz j4
l2:                                                                 ; will perform sar instruction
```

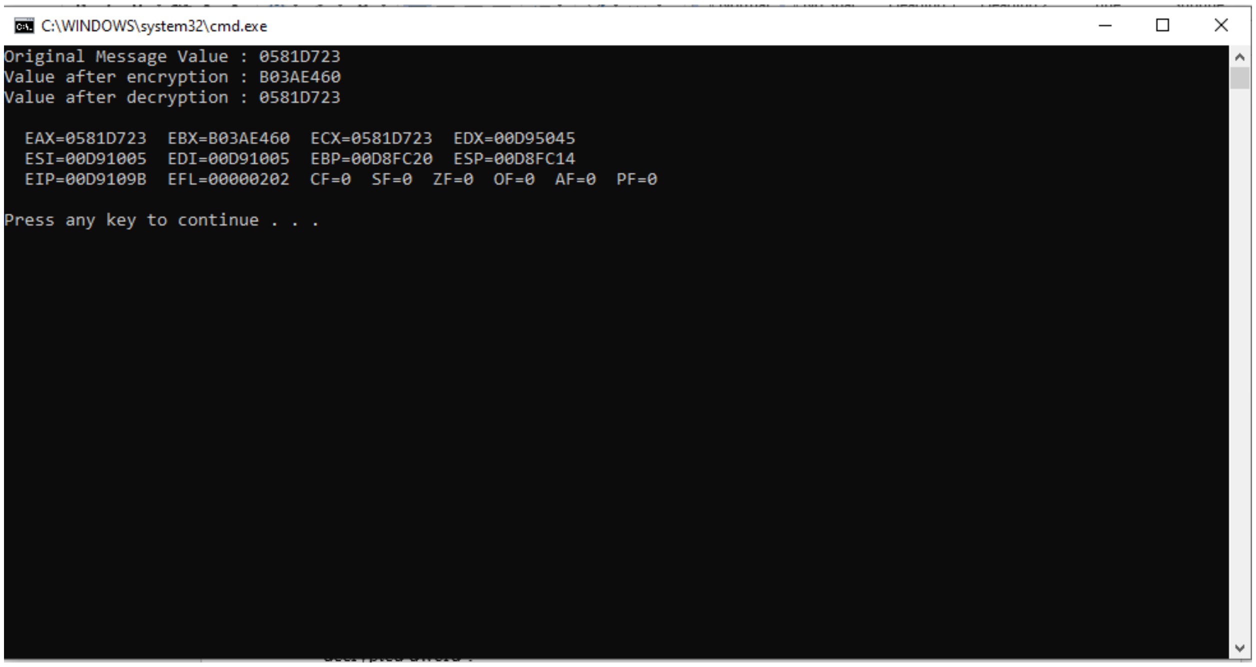
```
        test eax, 00000000000000000000000000000010b
        jnz last
        SAR eax, 1
loop l2


j4:
movzx ecx, count1
cmp ecx, 0
jz last                                          ; no loops afterwards so directly jump to last
l1:                                              ; will perform shr instruction
        test eax, 00000000000000000000000000000001b
        jnz last
        SHR eax, 1
loop l1

last:
mov decrypted, eax
ret

decrypt endp
end main
```

<u>OUTPUT SCREENSHOT:</u>



C:\WINDOWS\system32\cmd.exe

```
Original Message Value : 0581D723
Value after encryption : B03AE460
Value after decryption : 0581D723

  EAX=0581D723  EBX=B03AE460  ECX=0581D723  EDX=00D95045
  ESI=00D91005  EDI=00D91005  EBP=00D8FC20  ESP=00D8FC14
  EIP=00D9109B  EFL=00000202  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=0

Press any key to continue . . .
```