CS-235

Computer Organization and Assembly Language

Assignment 03

Name: Mahum Samar

Class: BSCS-9B

CMS ID: 290647

Submitted to: DR. Muhammad Imran

# Part 01

|  |
| --- |
| Score: 11/14 |

# Part 02

# Exercise # 01

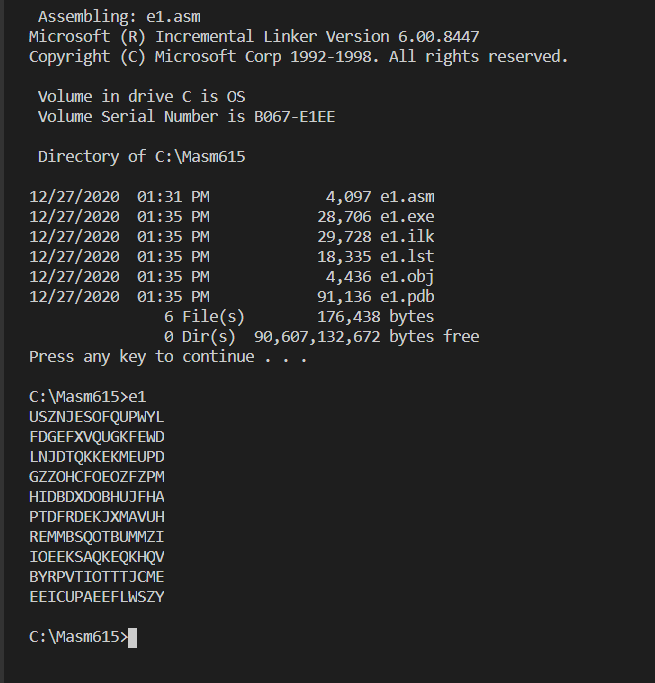
## Description:

## In this program the randomString procedure is called 10 times to generate the string of length 15 consisting of random capital letters. RandomString method users built-in function RandomRange to generate letters. 26 is moved in eax register to generate random numbers between zero to 25 and then randomRange method is called and then 26 is added to the random number generator because capital letters start from 65 in ASCII. The printArray procedure is used to print the array on the console.

## CODE:

|  |
| --- |
| ;-------------------------------------------------------------------------------------------------------------------------------------  TITLE String of length equal to lengthOfString with random capital letters  ;-------------------------------------------------------------------------------------------------------------------------------------  INCLUDE irvine32.inc  ; taking the length of the string as 15  lengthOfString = 15  ;-------------------------------------------------------------------------------------------------------------------------------------  .DATA  ;-------------------------------------------------------------------------------------------------------------------------------------  ; creating the array of same size of length to generate the random capital letters.  array BYTE lengthOfString DUP(?)  ;-------------------------------------------------------------------------------------------------------------------------------------  .CODE  ;-------------------------------------------------------------------------------------------------------------------------------------  main PROC  ; the length of the string is passed in the eax register  mov eax, lengthOfString  ; the offset to the array is passed in the esi register  mov esi, OFFSET array  ; loop is executed 10 times and passes in the ecx register  mov ecx,10  ; loop for generating the random array  randomArrayCall:  call RandomString ;random function called  loop randomArrayCall  exit  main ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  ; RandomString function is used to generate the array of string consisting if all the capital letters.  ; eax: length of array is passed in the eax register  ; eax, ecx, esi are pushed on the stack  ; the random array is generated and printed on the console  ;-------------------------------------------------------------------------------------------------------------------------------------  RandomString PROC uses eax ecx esi  mov ecx, eax  randomArray:    mov eax,26 ; 26 is moved to the eax register to genarte the random numbers between the range 0 to 26  call RandomRange ;build-in function to generate the random numbers  add eax, 65 ;65 added because capital letters start from 65 to 90  mov [esi], eax ;random number generated is moved to the offset where the pointer is present in the array  inc esi ;esi incremented  loop randomArray ;loop  call PrintArray ;method to print the array on screen  ret ;returns from procedure  RandomString ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  ; procedure to print the array on the screen  ; eax: length of array passed  ;-------------------------------------------------------------------------------------------------------------------------------------  PrintArray PROC uses esi eax ecx  ; esi: offset of array is passed  ; eax is moved to eax determinning how many times to execute the loop  mov esi, OFFSET array  mov ecx, eax  mov eax,0  pArray:  mov eax, [esi]  call writechar  inc esi  loop pArray  call crlf ;new line  ret  PrintArray ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  END main  ;------------------------------------------------------------------------------------------------------------------------------------- |

## OUTPUT:



# Exercise # 02

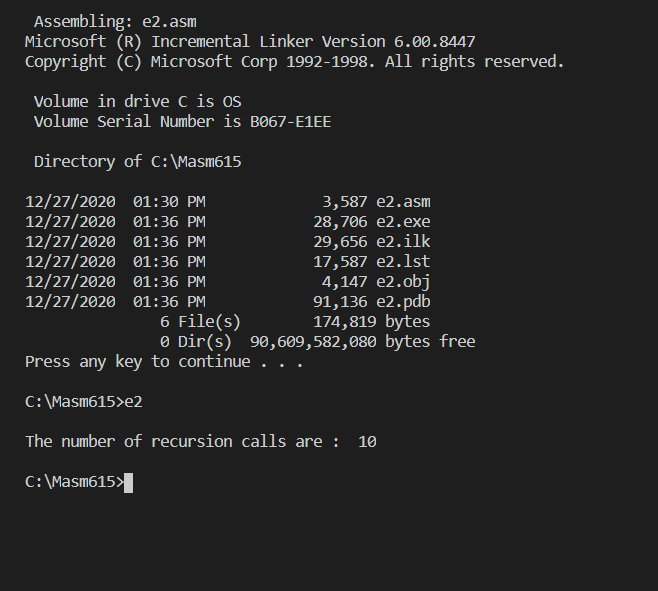
## Description:

This program is used for recursion calls 10 times . RecursionMethod is used for direct recursion. A variable is incremented by 1 each time method is called, loop statement is in start. So, that ECX is decremented before recursion and then it goes to the loop label and make the recursive call. After loop is finished i.e. ECX is 0, function returns call and after each recursiv call returns to the calling statement the function also return that's why two return statements are used to make the recursive calls for fixed number of times.

## CODE:

|  |
| --- |
| ;-------------------------------------------------------------------------------------------------------------------------------------  TITLE Recursion by using a loop statement without conditional statements  ;-------------------------------------------------------------------------------------------------------------------------------------  INCLUDE irvine32.inc  ;-------------------------------------------------------------------------------------------------------------------------------------  .DATA  ;-------------------------------------------------------------------------------------------------------------------------------------  recursionCount BYTE ? ;count number of times the recursion call is made  recursionCalled BYTE "The number of recursion calls are : ",0  ;-------------------------------------------------------------------------------------------------------------------------------------  .CODE  ;-------------------------------------------------------------------------------------------------------------------------------------  main PROC  ;-------------------------------------------------------------------------------------------------------------------------------------  call Crlf  mov recursionCount,0 ;0 moved to the recursion counter  mov ecx, 10 ;the recursion is called 10 times    call RecursionMethod ;method called  ;message printed to notify number of times recursion calls made  mov edx,OFFSET recursionCalled  call WriteString  ;after returning from the recursion call counter value is moved in eax to print it on the screen  mov eax, 0  mov al, recursionCount  call WriteDec ;writeDec to print on the console  call Crlf  exit  main ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  ; this method makes the recusive calls equal to the value moved in the ecx register  ; recursionCount variable is incremented each time the method is called to keep track how many times the method is called.  ;when ecx becomes zero the method returns and other recursion calls for the values are ommited.  ; thats why 2 returns are made otherwise the method is called  ; the number of times equal to n-1 times for each value of n because the loop is used.  ;-------------------------------------------------------------------------------------------------------------------------------------  RecursionMethod PROC  ;the counter is incremented each time the recursion call is made  inc recursionCount  ;firstly the loop is called so that the ecx is decremented before each iteration  loop recursionLoop  ;after the loop is finished one time the pointer returns from the procedure  ret  recursionLoop:  ;after coming to the label the recursive call is made  call RecursionMethod  ; after returning from the recursive call the procedure returns  ret  RecursionMethod ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  END main  ;------------------------------------------------------------------------------------------------------------------------------------- |

## OUTPUT:



# Exercise # 03

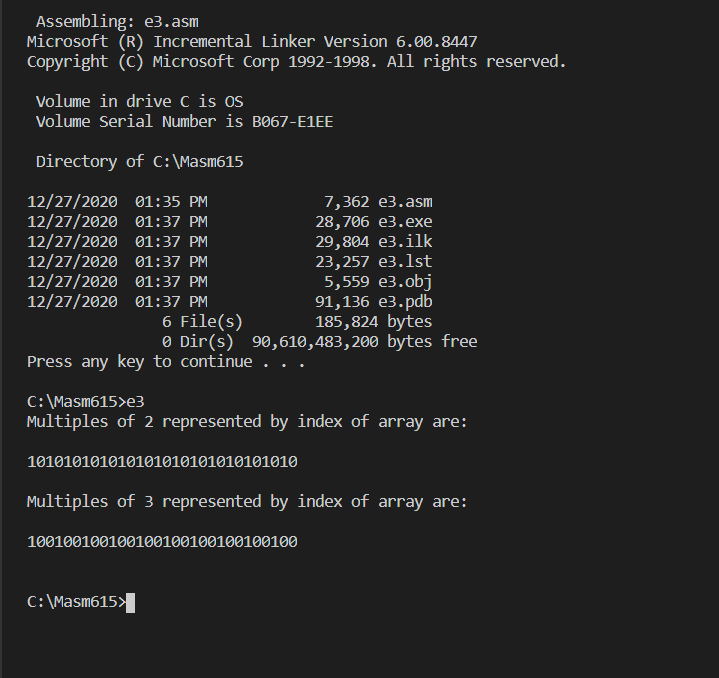
## Description:

In exercise 3. ‘N’ is assumed to be multiple of ‘K’. N is fixed in this program that is equal to 30. The array is created of size N and the value of two is moved in K before calling the function multipleOfK and N is moved in ECX register. EDX is used to move 1 in array to the corresponding multiple of the given number and ESI used to access the array position. The multipleLoop is executed N times and number whose multiple have to be found is subtracted from the current loop counter, it gives index position of array. One is moved to that position of array and counter is decremented by K - 1 to go to the previous multiple. The loop executes in descending order of multiples. ValueSub variable is equal to ‘K – 1’ because ECX is automatically decremented by 1 in loop statement. InitializeArray procedure is used to initialize the array each time the value of K is changed. PrintArray method is used to print the array on the console.

## CODE:

|  |
| --- |
| ;-------------------------------------------------------------------------------------------------------------------------------------  TITLE Finding multiples of number 'K' which are less than 'N' and filling '1' in corresponding array index  ;-------------------------------------------------------------------------------------------------------------------------------------  INCLUDE irvine32.inc  ;-------------------------------------------------------------------------------------------------------------------------------------  ;N is assumed that it is the multiple of the given number whose multiples are to be found.  ;This program is made using this assumption.  N = 30 ;the range until which the nultiples are to be found  ;-------------------------------------------------------------------------------------------------------------------------------------  .DATA  ;-------------------------------------------------------------------------------------------------------------------------------------  K BYTE ? ;specifies the number whose multiples are to be found  multipleTwo BYTE "Multiples of 2 represented by index of array are: ",0  multipleThree BYTE "Multiples of 3 represented by index of array are: ",0    array BYTE N dup(0) ;array created equal tot he size of the max range until which multiples are to be found  valueSub BYTE ? ;this variable is used to decrement the counter to find the previous multiple of the given number  ;-------------------------------------------------------------------------------------------------------------------------------------  .CODE  ;-------------------------------------------------------------------------------------------------------------------------------------  main PROC  mov ecx,0 ;initialized  mov cl,N ;loops N times  mov eax,0 ;for moving the number in K whose multiples are to be found.  mov al,2  mov K,al  mov valueSub,al ;1 less than the number whose multiples are to be found  call MultipleOfK ;method called to fill the index equal to the multiples of 'K' less than 'N'  mov edx, OFFSET multipleTwo  call WriteString  call Crlf  call Crlf  call PrintArray ;the array is printed      call initializeArray  mov ecx,0 ;initialized  mov cl,N ;loop N times  mov eax,0 ;for moving the number in K whose multiples are to be found.  mov al,3  mov K,al  mov valueSub,al ;1 less than the number whose multiples are to be found  call MultipleOfK ;method called to fill the index equal to the multiples of 'K' less than 'N'  mov edx, OFFSET multipleThree  call WriteString  call Crlf  call Crlf  call PrintArray ;the array is printed  exit  main ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  ; ecx is passed to notify the number of times the loop is to be executed  ; the array is modified at the index which are the multiplesof 'K'  ;multiples are found in descending order.  ; nothing is returned  ;-------------------------------------------------------------------------------------------------------------------------------------  MultipleOfK PROC  PUSHAD  mov edx,0 ;used to mov 1 to the index of array  mov dl,1  mov esi,0 ;for accessing the specific index of array  mov ebx,0 ;used to mov K so that can be subtracted to find previous multiple.    dec valueSub ;decremented by '1' to be subtracted from the ecx for next interation  multiple:  mov eax, 0  mov al,cl ;the iteration number moved to al  mov bl,K ;the number whose multiples are to be found  sub al,bl ;subtracting 'K' from Current multiple of 'K'  mov esi,eax ;the multiple found is used to make the corresponding index of array '1'  mov array[esi],dl ;'1' moved to index equal to the multiple  sub cl,valueSub ;'K-1' decremented because the cl is decremented itself in each iteration of loop  loop multiple  POPAD  ret ;returns  MultipleOfK ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  ;procedure to initialize the array for finding the next multiples for the number 'K'  ;-------------------------------------------------------------------------------------------------------------------------------------  initializeArray PROC USES edx esi ecx  mov esi,0 ;for accessing the array elements  mov ecx,N  mov edx,0 ;for moving 0 to array  ;initializing the array again.  iArray:    mov array[esi],dl  inc esi  loop iArray  ret  initializeArray ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  ;procedure to print the array on the screen  ;-------------------------------------------------------------------------------------------------------------------------------------  PrintArray PROC USES esi ecx eax    mov esi,0 ;from 1st index  mov ecx,0  mov cl,N ;for looping N times  pArray:    mov eax, 0  mov al, array[esi]  call WriteDec  inc esi  loop pArray  call crlf ;new line  call crlf ;new line  ret  PrintArray ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  END main  ;------------------------------------------------------------------------------------------------------------------------------------- |

## OUTPUT:



# Exercise # 04

## Description:

The length of maximum input is fixed i.e. 128. PlainTextInput procedure is used to input the string from the user. The translatePlainText is used to translate the given text into the cipher text or decrypt it from ciphertext to plaintext. The multivalued key is used for encryption/decryption. The single string character is XORed with single character of key. When the key reaches to its maximum length it is initialized again. printText procedure is used to print the required text on the screen.

In this program, took help from the example discussed in the class regarding encryption/decryption and modified the translatePlainText procedure, according to the requirements of the question.

## CODE:

|  |
| --- |
| ;-------------------------------------------------------------------------------------------------------------------------------------  TITLE Encryption and Decryption by using multivalued KEY  ;-------------------------------------------------------------------------------------------------------------------------------------  INCLUDE irvine32.inc  ;-------------------------------------------------------------------------------------------------------------------------------------  ;max input size from the user  MAXINPUT = 128  ;-------------------------------------------------------------------------------------------------------------------------------------  .DATA  ;-------------------------------------------------------------------------------------------------------------------------------------  KEY BYTE "ABXmv#7",0 ;key used as a string  inputPrompt BYTE "Enter the string : ",0 ;prompt the user to input the string  cipherText BYTE "Cipher text : ",0 ;specifies cipher text  decryptText BYTE "Decrypt text : ",0 ;specifies decryp ted text  plainText BYTE MAXINPUT+1 dup(0) ;saves the input text from the user, + 1 to save the null character  plainTextSize DWORD ? ;saves the size of input from the user  ;-------------------------------------------------------------------------------------------------------------------------------------  .CODE  ;-------------------------------------------------------------------------------------------------------------------------------------  main PROC  call plainTextInput ;call procedure to take input from the user  call TranslatePlainText ;converts plain text into cipher text  mov edx, OFFSET cipherText  call PrintText ;displays the encrypted text  call TranslatePlainText ;converts cipher text into the plain text  mov edx, OFFSET decryptText  call PrintText ;displays plain/decrypt text  EXIT  main ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  ; edx: takes offset of message to be printed before printing cipher/decrypt text  ; all the registers are pushed into the stack so that can be restored at the end of procedure using PUSHAD/POPAD  ;takes input from the user and store it in the plainText variable  ;moves size of input text in the plainTextSize variable  ;-------------------------------------------------------------------------------------------------------------------------------------  plainTextInput PROC  PUSHAD  mov edx, OFFSET inputPrompt  call writestring ;propts for input  mov ecx, MAXINPUT ;for specifying max input size  mov edx, OFFSET plainText  call readstring ;takes input  mov plainTextSize, eax ;moves size of input from eax to plainTextSize  call crlf ;new line  POPAD  ret ;return  plainTextInput ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  ;prints the text on the screen  ;it stores all the registers in stack before modifying it. and pops at the end of procedure  ;-------------------------------------------------------------------------------------------------------------------------------------  PrintText PROC  PUSHAD  call writestring ;prints the message whether cipher or decrypt text  mov esi, 0 ;moves '0' so that string index one is accessed  mov ecx, plainTextSize ;loops until end of string is reached  ptext:  mov eax, 0 ;for copying current index referenced  mov al, plainText[esi]  call WriteChar ;prints the character  inc esi ;incremenst esi  loop ptext ;loops  call crlf ;new line  call crlf  POPAD  ret  PrintText ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  ;procedure to translate the input text.  ; It a general method using XOR operator for encryption/decryption  ;all registers stored in stack and poped at the end  ;-------------------------------------------------------------------------------------------------------------------------------------  TranslatePlainText PROC  PUSHAD  mov esi,0 ;to access 1st index of text  mov ecx, 0  mov ecx, plainTextSize ;loops until end of string reached  keyInitialize:    mov edi,0 ;counter for key  translateText:  mov ebx,0 ;for moving current character of string to be encrypted/decrpted  mov bl, KEY[edi]  xor plainText[esi], bl ;XORing plain text character with the corresponding index of the key  inc esi ;increments string index  inc edi ;increments key index  cmp edi,7 ;compares the key index with max length of the key so that if the end of key is reached it can be initialized again  je keyInitialize ;jumps to the label where key is initialized again    loop translateText  POPAD  ret ;returns  TranslatePlainText ENDP  ;-------------------------------------------------------------------------------------------------------------------------------------  END main  ;------------------------------------------------------------------------------------------------------------------------------------- |

## OUTPUT:

Text

Description automatically generated