**COAL LAB 12 (LAB TASKS)**

**TASK # 01:**

**CODE:**

Title Task1 (test.asm)

include irvine32.inc

Scan\_String proto string1: ptr byte, size1:dword

.data

Str1 BYTE "127&amp;j~3#^&amp;\*#\*#45^",0

NotFound byte "# NOT FOUND",0

Found byte "CHARACTER '#' FOUND AT INDEX : ",0

sizeString dword ?

.code

main proc

mov ecx, lengthof Str1

mov sizeString, ecx

invoke Scan\_String, addr Str1, SizeString

call dumpregs

exit

main endp

Scan\_String proc, string1: ptr byte, size1: dword

mov edi, string1

mov ecx, size1

mov al, '#'

cld

repne scasb

jnz quit

dec edi

mov edx, offset Found

call WriteString

mov ebx, 26

sub ebx, ecx

mov eax, ebx

call writeDec

call crlf

jmp outside

quit:

mov edx, offset NotFound

call writeString

call crlf

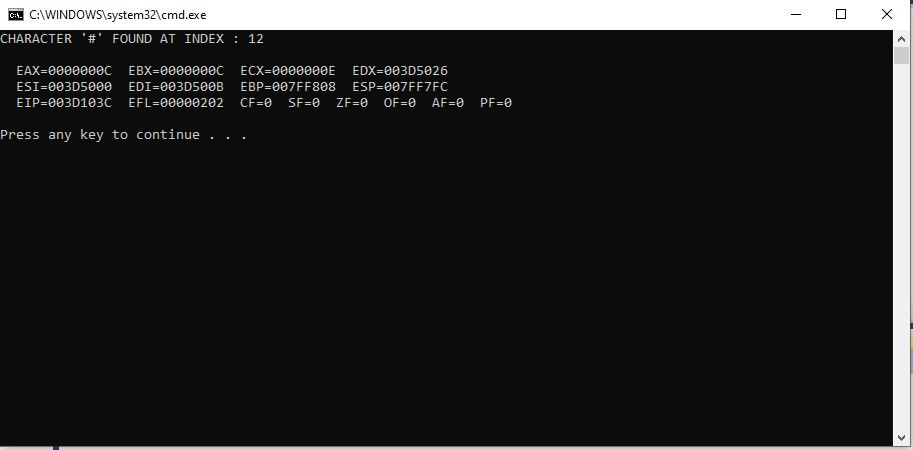
outside:

ret

Scan\_String endp

end main

**OUTPUT:**

****

**TASK # 02:**

**CODE:**

Title Task2 (test.asm)

include irvine32.inc

Scan\_String proto string1: ptr byte, char: ptr byte, size1:dword

.data

Str1 BYTE "127&amp;j~3#^&amp;\*#\*#45^",0

NotFound byte "# NOT FOUND",0

Found byte "CHARACTER FOUND AT INDEX : ",0

sizeString dword ?

Original byte "Original string : ",0

AskInput byte "ENTER THE CHARACTER TO SEARCH: ",0

inputChar byte ?

.code

main proc

mov ecx, lengthof Str1

mov sizeString, ecx

mov edx, offset Original

call WriteString

mov edx, offset Str1

call WriteString

call crlf

mov edx, offset AskInput

call WriteString

call readChar

mov inputChar, al

mov edx, offset inputChar

call WriteString

call crlf

invoke Scan\_String, addr Str1, addr inputChar, SizeString

call dumpregs

exit

main endp

Scan\_String proc, string1: ptr byte, char: ptr byte, size1: dword

mov edi, string1

mov ecx, size1

mov esi, char

mov al, [esi]

cld

repne scasb

jnz quit

dec edi

mov edx, offset Found

call WriteString

mov ebx, 26

sub ebx, ecx

mov eax, ebx

call writeDec

call crlf

jmp outside

quit:

mov edx, offset NotFound

call writeString

call crlf

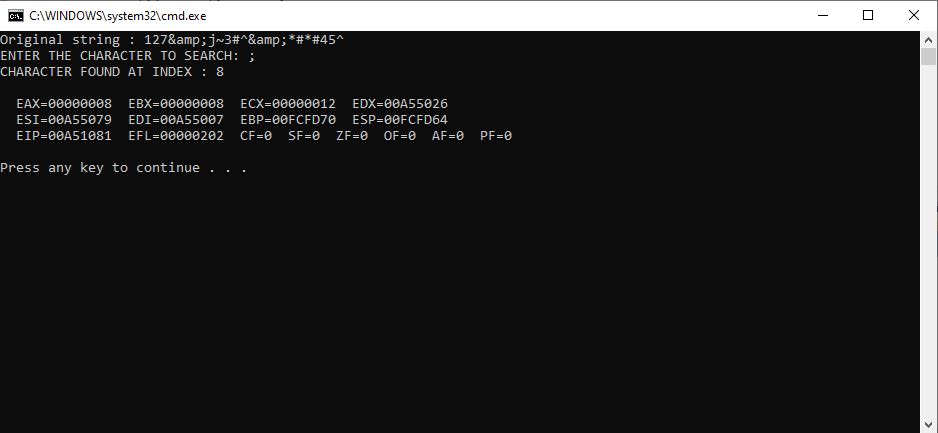
outside:

ret

Scan\_String endp

end main

**OUTPUT:**



**TASK # 03:**

**CODE:**

Title Task3 (test.asm)

include irvine32.inc

IsCompare proto string1: ptr byte, string2: ptr byte

.data

Source BYTE "abcdeg",0

Target Byte "adcdeg",0

Original Byte "Original Strings!",0

Matched Byte "These two strings matched.",0

UnMatched Byte "These two strings does not match.",0

.code

main proc

mov edx, offset Original

call WriteString

call crlf

mov edx, offset Source

call writeString

call crlf

mov edx, offset Target

call writeString

call crlf

invoke IsCompare, addr Source, addr Target

call dumpregs

exit

main endp

IsCompare proc, string1: ptr byte, string2: ptr byte

mov esi, string1

mov edi, string2

cmpsd

JA NotEqual

mov edx,offset Matched

call writestring

call crlf

jmp outside

NotEqual:

mov edx,offset UnMatched

call writestring

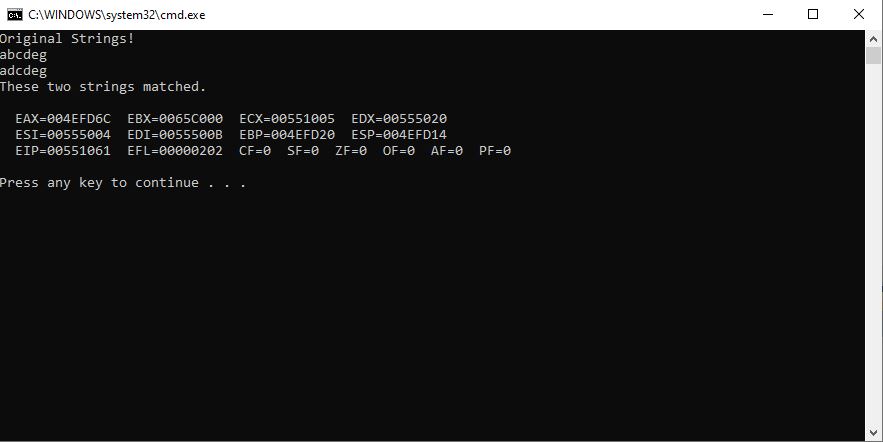
outside:

ret

IsCompare endp

end main

**OUTPUT:**



**TASK # 04:**

**CODE:**

Title Task3 (test.asm)

include irvine32.inc

Move proto string1: ptr byte, string2: ptr byte, n:dword

.data

Str1 byte "This is Ashmal Vayani",0

Str2 byte "I am in third semester",0

size1 dword ?

Original byte "Strings before moving : ",0

NewStrings byte "Strings after moving : " ,0

.code

main proc

mov ecx, lengthof Str1

mov size1, ecx

mov esi, offset Str1

mov edi, offset Str2

mov edx, offset Original

call WriteString

call crlf

mov edx, offset Str1

call WriteString

call crlf

mov edx, offset Str2

call WriteString

call crlf

call crlf

invoke Move, addr Str1, addr Str2, size1

call dumpregs

exit

main endp

Move proc, string1: ptr byte, string2: ptr byte, n: dword

cld

mov ecx, n

mov esi, string1

mov edi, string2

rep movsb

mov edx, offset NewStrings

call WriteString

call crlf

mov edx, offset Str1

call WriteString

call crlf

mov edx, offset Str2

call WriteString

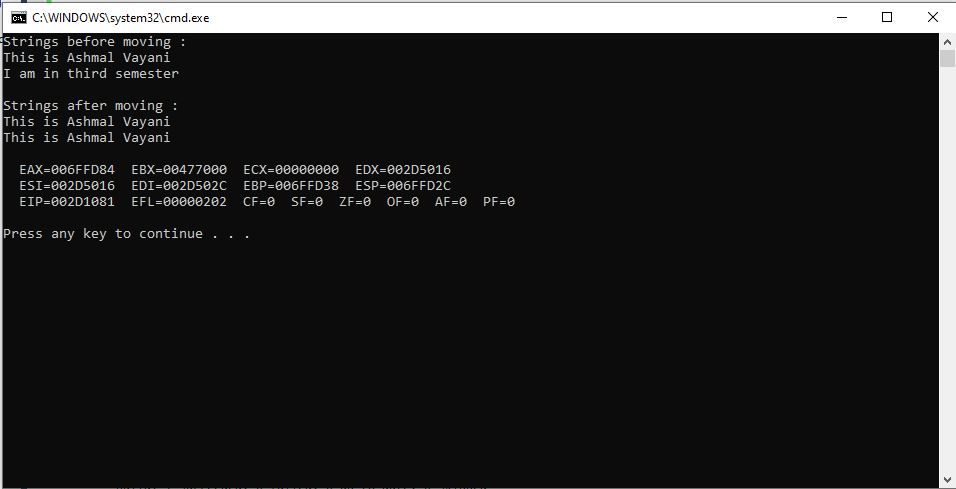
call crlf

ret

Move endp

end main

**OUTPUT:**



**TASK # 05:**

**CODE:**

Title Task5 (test.asm)

include irvine32.inc

Str\_Reverse proto string1: ptr byte, string2: ptr byte, size1:dword

.data

Str1 BYTE "Reverse this string.",0

Str2 byte "Hello world",0

Original BYTE "STRING TO BE REVERSED: ",0

Output BYTE "STRING AFTER REVERSING: ",0

SizeString dword ?

.code

main proc

mov ecx, lengthof Str1

mov SizeString, ecx

mov edx, offset Original

call WriteString

mov edx, offset Str1

call WriteString

call crlf

invoke Str\_Reverse, addr Str1, addr Str2, SizeString

call dumpregs

exit

main endp

Str\_Reverse proc, string1: ptr byte, string2: ptr byte, size1: dword

mov esi, string1

mov edi, string2

mov ecx, size1

mov edx, offset Output

call WriteString

mov eax, 0

mov ebx, ecx

sub ebx, 2

l1: mov al, [esi+ebx]

mov [edi], al

inc edi

dec ebx

loop l1

mov edx, offset Str2

call WriteString

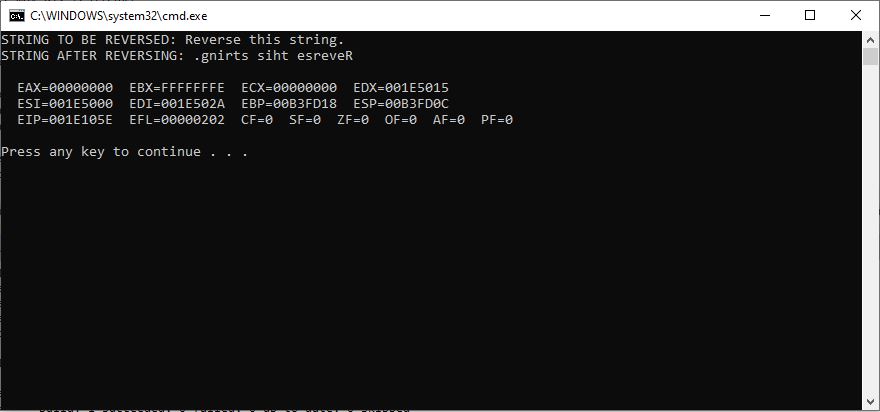
call crlf

ret

Str\_Reverse endp

end main

**OUTPUT:**



**TASK # 06:**

**CODE:**

Title Task6 (test.asm)

include irvine32.inc

Load\_Multiplier proto indexes: ptr dword, number: dword, size1: dword

.data

arrayIndexes dword 1,2,3,4,5,6,7,8,9,10

tableOf dword 7

SizeString dword ?

.code

main proc

mov ecx, lengthof arrayIndexes

mov SizeString, ecx

invoke Load\_Multiplier, addr arrayIndexes, addr tableOf, SizeString

call dumpregs

exit

main endp

Load\_Multiplier proc indexes: ptr dword, number: dword, size1: dword

mov esi, indexes

mov edi, indexes

mov ecx, size1

mov edx, 0

table:

lodsd

mul tableOf

stosd

loop table

mov ecx, size1

mov esi, indexes

l1: mov eax, [esi]

call WriteDec

call crlf

add esi, 4

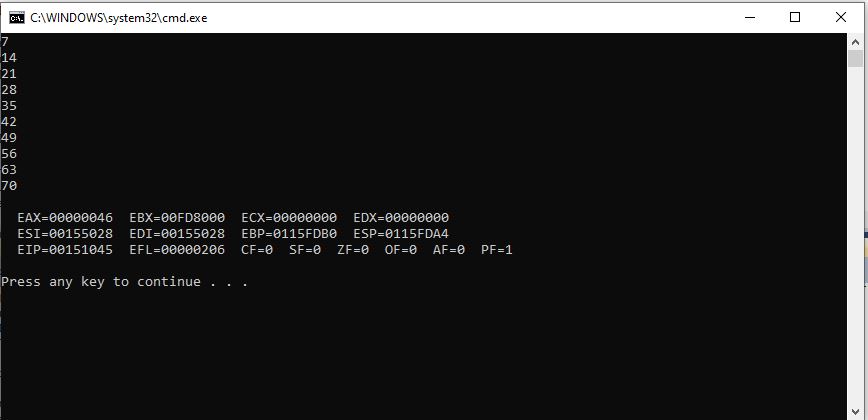
loop l1

ret

Load\_Multiplier endp

end main

**OUTPUT:**



**TASK # 07:**

**CODE:**

TITLE Task7 (test.asm)

include irvine32.inc

Binary\_Search PROTO, searchValue:DWORD, arrayPTR:PTR DWORD, count:DWORD

.data

arrayIndexes dword 1,3,5,7,9,11,13,15,17,19,21

key dword ?

total dword ?

Matched byte "The key value 13 found!",0

UnMatched byte "The key value was not found",0

space byte " ",0

Original byte "The original array : ",0

.code

main proc

mov key, 7

mov ecx, lengthof arrayIndexes

mov total, ecx

mov esi, offset arrayIndexes

mov ecx, lengthof arrayIndexes

mov edx, offset original

call WriteString

l1: mov eax, [esi]

call WriteDec

mov edx, offset space

call WriteString

add esi, 4

loop l1

mov ecx, lengthof arrayIndexes

call crlf

invoke Binary\_Search, key, addr arrayIndexes, ecx

cmp eax, -1

JNE found

mov edx, offset UnMatched

call writeString

call crlf

jmp Outside

found:

mov edx, offset Matched

call writeString

call crlf

Outside:

call dumpregs

exit

main endp

Binary\_Search proc USES edi, searchValue:DWORD, arrayPTR:PTR DWORD, count:DWORD

LOCAL first:DWORD, last:DWORD, mid:Dword ; local directives for local variable

mov first, 0

mov eax, count

dec eax

mov last, eax

mov edi, searchValue ; edi stores the value of key

mov ebx, arrayPTR ; ebx stores the offset of array

whileLoop: mov eax, first ; eax stores first

cmp eax, last

jg L5 ; only if l<=r, else exit

; calculating mid in eax and storing it in mid variable (local)

mov eax, last

add eax, first

mov cx, 2

mov edx, 0

div cx

mov mid,eax

; assigning values[mid] to edx

mov eax, mid

mov cx, 4

mov edx, 0

mul cx

mov esi, eax

mov edx,[ebx+esi]

; check if value is lesser

cmp edx,edi

jge L2

mov eax,mid

inc eax ; changing the value of first to mid+1

mov first,eax ; updating the value of first

jmp L4

; check if value is greater

L2:

cmp edx,edi

jle L3

mov eax,mid

dec eax ; changing last value to mid-1

mov last,eax

jmp L4

; if key == arr[mid] then return the value of mid while storing it in eax

L3:

mov eax,mid

jmp L9

L4:

jmp whileLoop ; loop again

; if value does not exists then assign -1 to eax and return to main

L5:

mov eax,-1

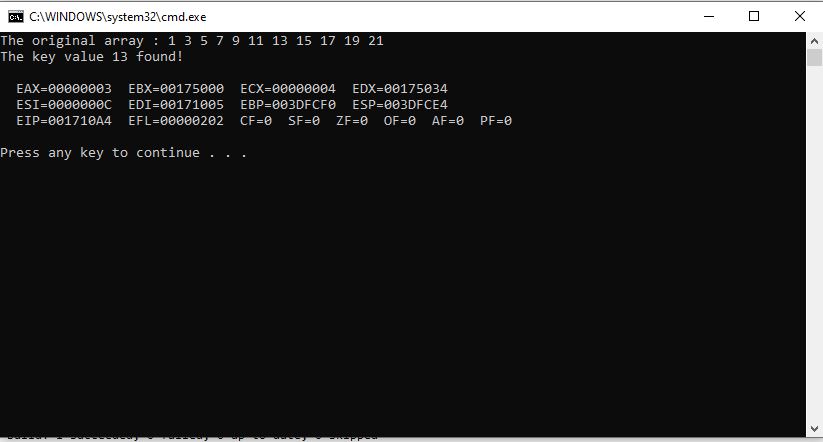
L9:

ret

Binary\_Search ENDP

END main

**OUTPUT:**



**TASK # 08:**

**CODE:**

Title Task8 (test.asm)

include irvine32.inc

Get\_frequencies proto table1: ptr byte, table2: ptr dword

.data

target BYTE "AAEBDCFBBC",0

freqTable DWORD 256 DUP(0)

TargetString byte "Target String : ", 0

Output byte "Frequency Table : ",0

space byte " ",0

.code

main proc

mov edx, offset TargetString

call WriteString

mov edx, offset target

call WriteString

call crlf

INVOKE Get\_frequencies, ADDR target, ADDR freqTable

call dumpregs

exit

main endp

Get\_frequencies proc, table1: ptr byte, table2: ptr dword

mov esi, table1

mov edi, table2

mov ecx, lengthof target

l1: mov eax, 0

mov al, [esi]

mov bl, 4

mov edx, 0

mul bl

mov bl, 1

add [edi+eax], bl

inc esi

loop l1

mov edx, offset Output

call WriteString

mov esi, offset freqTable

mov ecx, 11 ; because only required to print till 4B (11 characters), though we can extend it for 256 characters

add esi, 260 ; in this case we are starting from A, if it was different strings, then we have to start esi from base offset

mov edx, offset space

l2: mov eax, [esi]

call writeDec

call WriteString

add esi, 4

loop l2

call crlf

ret

Get\_frequencies endp

end main

**OUTPUT:**

