

Name: Owais Ali Khan  
Section: 3-F  
Roll: 21K-3298

Date:

## COAL GRAND ASSIGNMENT:-

### QUESTION #01:-

(i)

B9 00 12

=  $\begin{array}{ccc} 1011 & 1001 & 0012 \\ \text{MOV} & \text{16-bit CX} & \text{Disp} \end{array}$

= MOV CX, 1200h

8C 85 DC 01

= 1000 1100 DC 01

Not an instruction.

(ii)

MOV [SI+490], SP

OPCODE = 100010

D = 0 (Source is register)

W = 1 (16-bit)

MOD = 10 (Memory, 16-bit displacement)

Fig No. ☐

Date: \_\_\_\_\_

$\Rightarrow$  PUSH AX

OPCODE = 01010

Reg = 000

=  $\frac{01001}{5}, \frac{0000}{0}$

[ = 50 h ]

$\Rightarrow$  MOV AX, VAR+6 ; VAR = 0002h

Sol

OPCODE = 100010

D = 1

W = 1

MOD = 00

Reg ~~AX~~ = 000

R/M ~~AX~~ = 110 (Direct Address)

=  $\frac{1000}{8}, \frac{1011}{8}, \frac{0000}{0}, \frac{0110}{6}, 02\ 00$

[ = 8B 06 02 00 h ]

$\Rightarrow$  SUB CX, VAR2 ; VAR2 = 0008h

OPCODE = 001010

D = 1

W = 1

MOD = 00

Reg = 001

R/M = 110 (Direct Address)

Date: \_\_\_\_\_

Reg = 100

R/M = 100

$$= \underbrace{1000}_{8} \underbrace{1001}_{9} \underbrace{10100100}_{A4}, \quad 90 \ 04 \ h$$

$$[= 89A49004 \ h]$$

$\Rightarrow$  ADD AL, [BX+SI]

OPCODE = 000000

D = 1 (source is memory)

W = 0 (8-bit)

MOD = 00

Reg = 000

R/M = 000

$$= \underbrace{000}_{0} \underbrace{000}_{2} \underbrace{10}_{0} \underbrace{00}_{0} \underbrace{000}_{0} \underbrace{000}_{0}$$

$$[= 0200 \ h]$$

$\Rightarrow$  JNZ NEXT ; NEXT = 0008h

OPCODE = 0000 1111 1000

CCCC = 0101

$$= \underbrace{0000}_{0} \underbrace{1111}_{F} \underbrace{1000}_{8} \underbrace{0101}_{5} \overbrace{08 \ 00}^{\text{Disp}}$$

$$= 0F850800 \ h$$

Date:

= 0010, 1011, 0000, 1110, 0800h  
2 B 0 E

[= 280E0800h]

=> INC DX

Sol

OPCODE = 111111

W = 1

MOD = 11

R/M = 0010

= 1111, 1111, 1100, 0010  
F F C 2

[= FFC2h]

(iii)

a) AL = 85h

AL = 01111010

AL = 10000101

b) AL = 34h

3D = 00111101

74 = 01110100

c) AL = BFh

AND = 00110100

d) AL = AEh

9B = 10011011

35 = 00110101

OR = 10111111

72 = 01110010

DC = 11011100

XOR = 10101110



Date: \_\_\_\_\_

(iv)

Ans) The first instruction moves the memory address of VAR1 into eax, while the second one moves the value of VAR1 into eax.

(v)

- 1) To save return address of procedures.
- 2) To pass arguments.
- 3) To save register values.
- 4) To create local variables.
- 5) To ~~return~~ save loop counters.

(vi)

EAX = 1234h, EBX = 5678h, ECX = 9ABCh, ESP = 0100h

After execution:-

EAX = 9ABCh

EBX = 9ABCh

ECX = 5678h

ESP = 0100h

(vii)

Generated instructions:-

MYSUM PROC

PUSH ESI

PUSH ECX

...

POP ECX

POP ESI

ret

Date: \_\_\_\_\_

~~WAV~~ (ix)

	AH = 1	AH = 2	AH = 3
If $V1 = V2$			✓
If $V1 < V2$		✓	
If $V1 > V2$	✓		

(~~viii~~)

Segment	Start	Stop	Length
Code (CS)	508A0h	5099Fh	00100h
Data (DS)	509A0h	509EFh	00050h
Stack (SS)	509F0h	50BEFh	00100h

Date: \_\_\_\_\_

(x) / (iii)

ESP = 00001FF8h

00001FF8h

~~ESP =~~

EBP = 00001FE2h

00001FF8

6H

00001FF4

5H

EAX = 0Bh

00001FF0

Return Address

00001FE6

EBP

ESP: 00001FE2

ESP = 00001FDAh

00001FF8

6H

00001FF4

5H

00001FF0

Return Address

00001FE6

EBP

00001FE2

OFFSET X1

00001FDE

OFFSET X2

00001FDA

X2 = 32h

X1 = 30h

EIP = 115000000h



Date: \_\_\_\_\_

## QUESTION#02:-

- (i) EAX = 2h
- (ii) EAX = 9h
- (iii) EAX = 30h
- (iv) EAX = 40400Ch
- (v) EAX = 0A302010h
- (vi) AL = ~~0A302010~~ 0A302010h
- (vii) Yes, zero flag will be set, because ax contains 0h.
- (viii) Yes, it is possible for neg to set overflow flag.
- (ix) Invalid; I is a constant.
- (x) Valid; AL = 38
- (xi) Valid; EAX = 00006534h
- (xii) Valid; EAX = 0091FE50h
- (xiii) Invalid; Need to give size of [ESI]
- (xiv) Invalid; Cannot have two memory operands.
- (xv) Valid; EAX = 3938386h
- (xvi) Invalid; Invalid operands
- (xvii) Invalid; operands, not of same size
- (xviii) AX = FFF6
- (xix)
  - 1) d
  - 2) e
  - 3) c
  - 4) a
  - 5) b

(xx)

AL = 6Ah  
AL = EAh



Date: \_\_\_\_\_

(xi)

CF = 0  
SF = 1  
ZF = 0  
OF = 0  
AX = F2EBh

(xii)

CF = 1  
OF = 0  
SF = 0  
ZF = 0  
CX = 16C0h

(xiii)

CF = 1  
SF = 1  
OF = 0  
ZF = 0  
BX = FFFEh

(xx) EAX = 0000FFFF

(xxi) AND EAX, 0F0F0000h

(xxii) \*cmp 7FFh, 8000h

JB would execute because it takes unsigned values  
JL would not execute because it takes signed values.

## Question # 03(i):

TITLE QUESTION 3(i)  
INCLUDE Irvine32.inc  
INCLUDE Macros.inc

.data  
OP1 SDWORD ?  
OP2 SDWORD ?  
OP3 SDWORD ?  
X DWORD 0  
Y DWORD 0  
VAL1 SDWORD ?  
VAL2 SDWORD ?  
VAL3 SDWORD ?

.code  
main PROC  
WHILE:  
    mov eax, OP1  
    cmp eax, OP2  
    jge END\_WHILE  
    mov eax, OP3  
    cmp eax, OP2  
    jne ELSE  
    mov ebx, Y  
    add ebx, 2  
    mov X, ebx  
    ELSE:  
        mov ebx, Y  
        add ebx, 10  
        mov X, ebx  
    jmp WHILE

exit  
main ENDP

main PROC  
mov eax, VAL1  
mov ebx, VAL2  
mov ecx, VAL3  
cmp eax, ebx  
jle ENDD  
cmp ebx, ecx

```
jle ENDD
mov x, 10
ENDD:
    mov x, 20
exit
main ENDP
END main
```

## Question # 03(ii):

```
TITLE QUESTION 3(ii)
INCLUDE Irvine32.inc
INCLUDE Macros.inc
```

```
.data
X DWORD 21
Y DWORD 33
Z DWORD 61
```

```
.code
main PROC
mov eax, X
mov ebx, Y
mov ecx, Z
call MINIMUM
mWrite "Minimum value is: "
call WriteDec
call Crlf
exit
main ENDP
```

MINIMUM PROC ; takes arguments in eax, ebx and ecx. Returns minimum value in eax.

```
cmp eax, ebx
jle NEXT
mov eax, ebx
NEXT:
    cmp eax, ecx
    jle ENDD
    mov eax, ecx
ENDD:
    ret
MINIMUM ENDP

END main
```



## Question # 03(iii):

TITLE QUESTION 3(iii)

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

Table1 BYTE 100 DUP(?)

Table2 BYTE 100 DUP(?)

Constant DWORD ?

.code

main PROC

call Copy

push Constant

push OFFSET Table1

push LENGTHOF Table1

call Search

exit

main ENDP

Copy PROC

cld

mov esi, OFFSET Table1

mov edi, OFFSET Table2

mov ecx, LENGTHOF Table1

rep MOVSB

ret

Copy ENDP

Search PROC

MOV ecx, [esp + 4] ; Length

MOV esi, [esp + 8] ; Offset

MOV eax, [esp + 12] ; Value

MOV edi, -1

L1:

INC edi

CMP al, [esi + edi]

LOOPNZ L1

JZ L2

INC edi

L2:

ret 12

Search ENDP

END main

## Question # 03(iv-a):

TITLE QUESTION 3(iv)

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

n DWORD 5

k DWORD 2

.code

main PROC

mWrite "Enter value of n: "

call ReadDec

mov n, eax

mWrite "Enter value of k: "

call ReadDec

mov k, eax

call CrLf

push k

push n

call Binomial

mWrite "Binomial coefficient is: "

call WriteDec

call CrLf

exit

main ENDP

Binomial PROC

push ebp

mov ebp, esp

push ebx

sub esp, 8

mov eax, [ebp+12]

cmp eax, [ebp+8]

jle L2

mov eax, 0

jmp L3

L2:

cmp [ebp+12], 0

je L4

mov eax, [ebp+12]

cmp eax, [ebp+8]

```

        jne     L5
L4:     mov     eax, 1
        jmp     L3
L5:     mov     edx, [ebp+12]
        sub     edx, 1
        mov     eax, [ebp+8]
        sub     eax, 1
        mov     [esp+4], edx
        mov     [esp], eax
        call    Binomial
        mov     ebx, eax
        mov     edx, [ebp+8]
        sub     edx, 1
        mov     eax, DWORD PTR [ebp+12]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], edx
        call    Binomial
        add     eax, ebx
L3:     add     esp, 8
        pop     ebx
        pop     ebp
        ret

```

Binomial ENDP

END main

## Question # 03(iv-b):

TITLE QUESTION 3(iv.b)

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

x DWORD 2

n DWORD 4

.code

main PROC

mWrite "Enter value of x: "

call ReadDec

mov x, eax



```
mWrite "Enter value of n: "  
call ReadDec  
mov n, eax  
call CrLf  
push n  
push x  
call Power  
mWrite "Nth power of x is: "  
call WriteDec  
call CrLf
```

```
exit  
main ENDP
```

```
Power PROC  
push ebp  
mov ebp, esp  
sub esp, 8  
mov eax, [ebp + 12]  
cmp eax, 0  
jne L1  
mov eax, 1  
jmp ENDD  
L1:  
    mov edx, [ebp + 12]  
    sub edx, 1  
    mov eax, [ebp + 8]  
    mov [esp + 4], edx  
    mov [esp], eax  
    call Power  
    mov edx, 0  
    mul DWORD PTR [esp]  
ENDD:  
    add esp, 8  
    pop ebp  
    ret  
Power ENDP
```

```
END main
```

## Question # 03(v):

```
TITLE QUESTION 3(v)  
INCLUDE Irvine32.inc  
INCLUDE Macros.inc
```

```
.data
n DWORD 1
```

```
.code
main PROC
mWrite "Enter value of n: "
call ReadDec
mov n, eax
call Crlf
push n
call Fibonacci
mWrite "Nth term of Fibonacci sequence is: "
call WriteDec
call Crlf
```

```
exit
main ENDP
```

```
Fibonacci PROC
push ebp
mov ebp, esp
push ebx
sub esp, 4
mov eax, [ebp + 8]
cmp eax, 0
jne NEXT
mov eax, 0
jmp ENDD
NEXT:
    cmp eax, 1
    jne L1
    mov eax, 1
    jmp ENDD
L1:
    mov eax, [ebp + 8]
    sub eax, 1
    mov [esp], eax
    call Fibonacci
    mov ebx, eax
    mov eax, [ebp + 8]
    sub eax, 2
    mov [esp], eax
    call Fibonacci
    add eax, ebx
ENDD:
    add esp, 4
```

```
        pop ebx
        pop ebp
        ret
Fibonacci ENDP
```

```
END main
```

## Question # 03(vi):

```
TITLE QUESTION 3(vi)
INCLUDE Irvine32.inc
INCLUDE Macros.inc
```

```
.data
intArray DWORD 60, 4, 17, 45, 7, 69, 21, 33, 96, 81
count DWORD 10
```

```
.code
main PROC
mWrite "Array values: "
mov ecx, count
mov esi, OFFSET intArray
PRINT:
    mov eax, [esi]
    call WriteInt
    mWrite " "
    add esi, 4
    loop PRINT
call Crlf
call Crlf
mov ecx, count
mov esi, OFFSET intArray
call Exchange
mWrite "After Exchange Sort: "
mov ecx, count
mov esi, OFFSET intArray
PRINT2:
    mov eax, [esi]
    call WriteInt
    mWrite " "
    add esi, 4
    loop PRINT2
call Crlf
exit
main ENDP
```



Exchange PROC

mov eax, 0

mov ebx, 0

L1:

    cmp ecx, 1

    JBE END\_LOOP1

    push ecx

    lea edi, [esi+4]

    L2:

        cmp ecx, 1

        JBE END\_LOOP2

        mov eax, [esi]

        mov ebx, [edi]

        cmp eax, ebx

        JBE NotSwapped

        push edi

        push esi

        call SWAP

        pop esi

        pop edi

        NotSwapped:

            add edi, 4

            loop L2

    END\_LOOP2:

        add esi, 4

        pop ecx

        loop L1

END\_LOOP1:

    ret

Exchange ENDP

SWAP PROC

push ebp

mov ebp, esp

mov esi, [ebp + 8]

mov edi, [ebp + 12]

mov eax, [esi]

mov ebx, [edi]

xchg eax, ebx

mov [esi], al

mov [edi], bl

mov [ebp + 8], esi

mov [ebp + 12], edi

mov esp, ebp

pop ebp

```
ret
SWAP ENDP
```

```
END main
```

## Question # 03(vii):

```
TITLE Question 3(vii)
INCLUDE Irvine32.inc
INCLUDE Macros.inc
```

```
.data
intArray DWORD 60, 4, 17, 45, 7
count DWORD 5
```

```
.code
main PROC
mWrite "Array values: "
mov ecx, count
mov esi, OFFSET intArray
PRINT:
    mov eax, [esi]
    call WriteInt
    mWrite " "
    add esi, 4
    loop PRINT
call Crlf
call Crlf
push count
push OFFSET intArray
call SelectSort
mWrite "After SelectSort: "
mov ecx, count
mov esi, OFFSET intArray
PRINT2:
    mov eax, [esi]
    call WriteInt
    mWrite " "
    add esi, 4
    loop PRINT2
call Crlf
exit
main ENDP
```

```
SelectSort PROC, array:PTR DWORD, array_size:DWORD
```

```

mov esi, array
mov ecx, array_size
mov eax, 0
L1:
    push ecx
    push esi
    mov eax, [esi]
    mov edi, array
    L2:
        mov ebx, [edi]
        cmp eax, ebx
        JAE NotSwapped
        mov eax, ebx
        mov esi, edi
        NotSwapped:
            add edi, 4
            loop L2
    sub edi, 4
    push edi
    push esi
    call SWAP
    pop esi
    pop ecx
    loop L1
ret
SelectSort ENDP

```

```

SWAP PROC
push ebp
mov ebp, esp
mov esi, [ebp + 8]
mov edi, [ebp + 12]
mov eax, [esi]
mov ebx, [edi]

xchg eax, ebx
mov [esi], eax
mov [edi], ebx
mov [ebp + 8], esi
mov [ebp + 12], edi

mov esp, ebp
pop ebp
ret 8
SWAP ENDP

```

```

END main

```

## Question # 04(i):

```
TITLE Question 4(i)
INCLUDE Irvine32.inc
INCLUDE Macros.inc
```

```
.data
Sequence_Number DWORD 0
Revision_Count DWORD 0
Status DWORD 0
Sensor_Data DWORD 0
```

```
.code
main PROC
mov ebx, eax
AND ebx, 0FFFh
mov Sequence_Number, ebx
mov ebx, eax
shr ebx, 12
AND ebx, 111b
mov Revision_Count, ebx
mov ebx, eax
shr ebx, 15
AND ebx, 1b
mov Status, ebx
mov ebx, eax
shr ebx, 16
AND ebx, 0FFFFh
mov Sensor_Data, ebx
```

```
mWrite "EAX: "
call WriteBin
call Crlf
mWrite "Sequence_Number: "
mov eax, Sequence_Number
call WriteBin
call Crlf
mWrite "Revision_Count: "
mov eax, Revision_Count
call WriteBin
call Crlf
mWrite "Status: "
mov eax, Status
call WriteBin
```

```
call CrLf
mWrite "Sensor_Data: "
mov eax, Sensor_Data
call WriteBin
call CrLf
```

```
exit
main ENDP
END main
```

## Question # 04(ii):

```
TITLE Question 4(ii)
INCLUDE Irvine32.inc
INCLUDE Macros.inc
```

```
.data
X DWORD 0
```

```
.code
main PROC
mWrite "Enter value of X: "
call ReadInt
mov X, eax
mov ebx, X
mov ecx, X
mov edx, X
shl eax, 0
shl ebx, 1
shl ecx, 2
shl edx, 4
add eax, ebx
add eax, ecx
add eax, edx
call CrLf
mWrite "X multiplied by 23 is: "
call WriteInt
call CrLf
```

```
exit
main ENDP
END main
```

## Question # 04(iii):

Date: \_\_\_\_\_

(x) / (iii)

ESP = 00001FF8h

00001FF8h

~~ESP =~~

EBP = 00001FE2h

00001FF8

6H

00001FF4

5H

EAX = 0Bh

00001FF0

Return Address

00001FE6

EBP

ESP: 00001FE2

ESP = 00001FDAh

00001FF8

6H

00001FF4

5H

00001FF0

Return Address

00001FE6

EBP

00001FE2

OFFSET X1

00001FDE

OFFSET X2

00001FDA

X2 = 32h

X1 = 30h

EIP = 115000000h

## Question # 04(iv):

```
TITLE Question 4(iv)
INCLUDE Irvine32.inc
INCLUDE Macros.inc
```

```
.data
source BYTE "this is the source string.", 0
size_S = ($ - source)
target BYTE size_S DUP(0)
```

```
.code
main PROC
cld
mWrite "Source string: "
mov edx, OFFSET source
call WriteString
call Crlf
mov ebx, 0
mov esi, OFFSET source
mov ecx, size_S
L1:
    push ecx
    mov edi, OFFSET target
    lodsb
    mov ecx, size_S
    repne scasb
    jz CONTINUE
    mov target[ebx], al
    add ebx, 1
    CONTINUE:
        pop ecx
        loop L1
mov target[ebx], 0
mWrite "Target string: "
mov edx, OFFSET target
call WriteString
call Crlf

exit
main ENDP
END main
```

## Question # 04(v):



TITLE QUESTION 4(v)

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

intArray SDWORD 60, 4, 17, 45, 7, 69, 21, 33, 96, 81

count DWORD 10

integer SDWORD ?

;SEARCH\_RECURSE PROTO, array:PTR SDWORD, arrayLength:DWORD, value:SDWORD

;INVOKE SEARCH\_RECURSE, ADDR intArray, count, integer

.code

main PROC

mWrite "Array values: "

mov ecx, count

mov esi, OFFSET intArray

PRINT:

    mov eax, [esi]

    call WriteInt

    mWrite " "

    add esi, 4

    loop PRINT

call Crlf

call Crlf

mWrite "Enter integer value to search: "

call ReadInt

mov integer, eax

push integer

push count

push OFFSET intArray

call SEARCH\_RECURSE

cmp eax, -1

je NOT\_FOUND

mWrite "Given value: "

xchg eax, integer

call WriteDec

xchg eax, integer

mWrite " found at index: "

call WriteDec

call Crlf

jmp ENDD

NOT\_FOUND:

    mWrite "Given value not found in array."

    call Crlf

ENDD:

    exit

main ENDP

```

;SEARCH_RECURSE PROC, array:PTR SDWORD, arrayLength:DWORD, value:SDWORD
SEARCH_RECURSE PROC
push ebp
mov ebp, esp
sub esp, 12
mov ecx, [ebp + 12]
cmp ecx, 0
jle ENDD
mov esi, [ebp + 8]
mov eax, [esi]
cmp eax, [ebp + 16]
je FOUND
mov eax, -1
mov esi, [ebp + 8]
add esi, 4
mov ecx, [ebp + 12]
dec ecx
mov edx, [ebp + 16]

mov [esp + 8], edx
mov [esp + 4], ecx
mov [esp], esi
call SEARCH_RECURSE
jmp ENDD
FOUND:
    mov eax, 10
    sub eax, [ebp + 12]
    add esp, 12
    pop ebp
    ret
ENDD:
    add esp, 12
    pop ebp
    ret
SEARCH_RECURSE ENDP

END main

```

## Question # 04(vi):

```

TITLE QUESTION 4(vi)
INCLUDE Irvine32.inc
INCLUDE Macros.inc

```

```

.data

```

MOON BYTE 20 DUP(?)

.code

main PROC

call STAR\_ARRAY

mov esi, OFFSET MOON

mov ecx, 20

PRINT:

mov al, [esi]

call WriteChar

inc esi

loop PRINT

call CrLf

exit

main ENDP

STAR\_ARRAY PROC

push ebp

mov ebp, esp

sub esp, 20

mov ecx, 19

L1:

cmp ecx, 0

js L2

lea edx, [ebp-20]

mov eax, ecx

add eax, edx

mov BYTE PTR [eax], 42 ; '\*' = 42

mov eax, ecx

add eax, OFFSET MOON

mov BYTE PTR [eax], 120 ; 'x' = 120

sub ecx, 1

jmp L1

L2:

lea esi, [ebp-20]

mov ecx, 20

PRINT:

mov al, [esi + ecx - 1]

call WriteChar

loop PRINT

call CrLf

add esp, 20

pop ebp

ret

STAR\_ARRAY ENDP

END main

## Question # 04(vii):

TITLE QUESTION 4(vii)

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

dividend\_d DWORD 0D4A4h

divisor\_d DWORD 0Ah

DIVIDE PROTO, dividend:DWORD, divisor:DWORD

.code

main PROC

INVOKE DIVIDE, dividend\_d, divisor\_d

exit

main ENDP

DIVIDE PROC, dividend:DWORD, divisor:DWORD

mov eax, dividend

mov edx, 0

mWrite "Dividend = "

call WriteHex

call CrLf

div divisor

cmp eax, 05h

jbe BASE

mov dividend, eax

INVOKE DIVIDE, dividend, divisor

BASE:

ret

DIVIDE ENDP

END main

## Question # 04(viii):

TITLE QUESTION 4(viii)

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

```
ArraySearchValues BYTE 20 DUP(1)
ArrayValues BYTE 1000 DUP(1)
```

```
.code
main PROC
cld
mov esi, OFFSET ArraySearchValues
mov edi, OFFSET ArrayValues
lea ecx, [1000 - 20]
L1:
    push ecx
    push esi
    push edi
    mov ecx, 20
    repe cmpsb
    je FOUND
    pop edi
    pop esi
    pop ecx
    add edi, 1
    loop L1
mWrite "ArraySearchValues not found in ArrayValues."
call CrLf
jmp ENDD
FOUND:
    mWrite "ArraySearchValues found at "
;    mov eax, esi
    lea eax, [esi - 20]
    sub eax, OFFSET ArraySearchValues
    call WriteDec
    call CrLf
ENDD:
    exit
main ENDP

END main
```

## Question # 04(ix):

```
TITLE QUESTION 4(ix)
INCLUDE Irvine32.inc
INCLUDE Macros.inc
```

```
.data
string BYTE "SITYA", 0
```

```
source BYTE "FAST NATIONAL UNIVERSITY", 0
s_size = ($ - source) - 1
target DWORD s_size DUP(?)
```

```
.code
main PROC
mov eax, 0
mov esi, OFFSET source
mov edi, OFFSET target
mov ecx, s_size
L1:
    mov al, BYTE PTR [esi]
    mov DWORD PTR [edi], eax
    add esi, 1
    add edi, 4
    loop L1
```

```
mov edi, OFFSET target
mov ecx, s_size
PRINT:
    mov eax, DWORD PTR [edi]
    call WriteDec
    mWrite " "
    add edi, 4
    loop PRINT
```

```
call Crlf
call SEARCH
```

```
exit
main ENDP
```

```
SEARCH PROC
mov esi, OFFSET string
mov edi, OFFSET source
lea ecx, [s_size - 1]
L1:
    INVOKE Str_compare, esi, edi
    je FOUND
    add edi, 1
    loop L1
mWrite "Given substring: "
mov edx, OFFSET string
call WriteString
mWrite " not found in original string."
call Crlf
jmp ENDD
FOUND:
    mWrite "Found substring: "
```

```
    mov edx, OFFSET string
    call WriteString
    mWrite " in original string."
    call Crlf
```

ENDD:

```
    ret
```

SEARCH ENDP

END main