Name: Muhammad Haris

Roll No: 22k-0505

Section BCS-3k

Instructor: Miss Atiya Jhokio

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q1 Code:**

INCLUDE Irvine32.inc

INCLUDE macros.inc

.data

len dword ?

array sdword 100 dup(?)

array2 sdword 100 dup(?)

.code

print proto, arr:ptr sdword, arr2:ptr sdword, s:dword

main Proc

xor eax, eax

xor ebx, ebx

xor ecx, ecx

xor edx, edx

xor esi, esi

xor edi, edi

mwrite "Enter Size of your array "

call readint

mov len, eax

mov ecx, len

mov esi, offset array

mwrite " First Array "

call crlf

l1:

xor eax, eax

mwrite " Enter Element "

inc ebx

mov eax,ebx

call writedec

xor eax, eax

call readint

mov [esi], eax

add esi, 4

call crlf

loop l1

mov ecx, len

mwrite "Second Array"

mov edi, offset array2

call crlf

xor ebx,ebx

l2:

xor eax, eax

mwrite " Enter Element "

inc ebx

mov eax,ebx

call writedec

xor eax, eax

call readint

mov [edi], eax

add edi, 4

call crlf

loop l2

mov ecx, len

invoke print, addr array, addr array2, ecx

mwrite " Number of Elemnts matched "

call writedec

exit

main endp

print proc, arr1:ptr sdword, arr2:ptr sdword, s:dword

mov ecx, s

xor eax, eax

mov esi,arr1

mov edi,arr2

l3:

mov ebx,[esi]

cmp [edi], ebx

je go

jmp last

go:

inc eax

last:

add esi, 4

add edi, 4

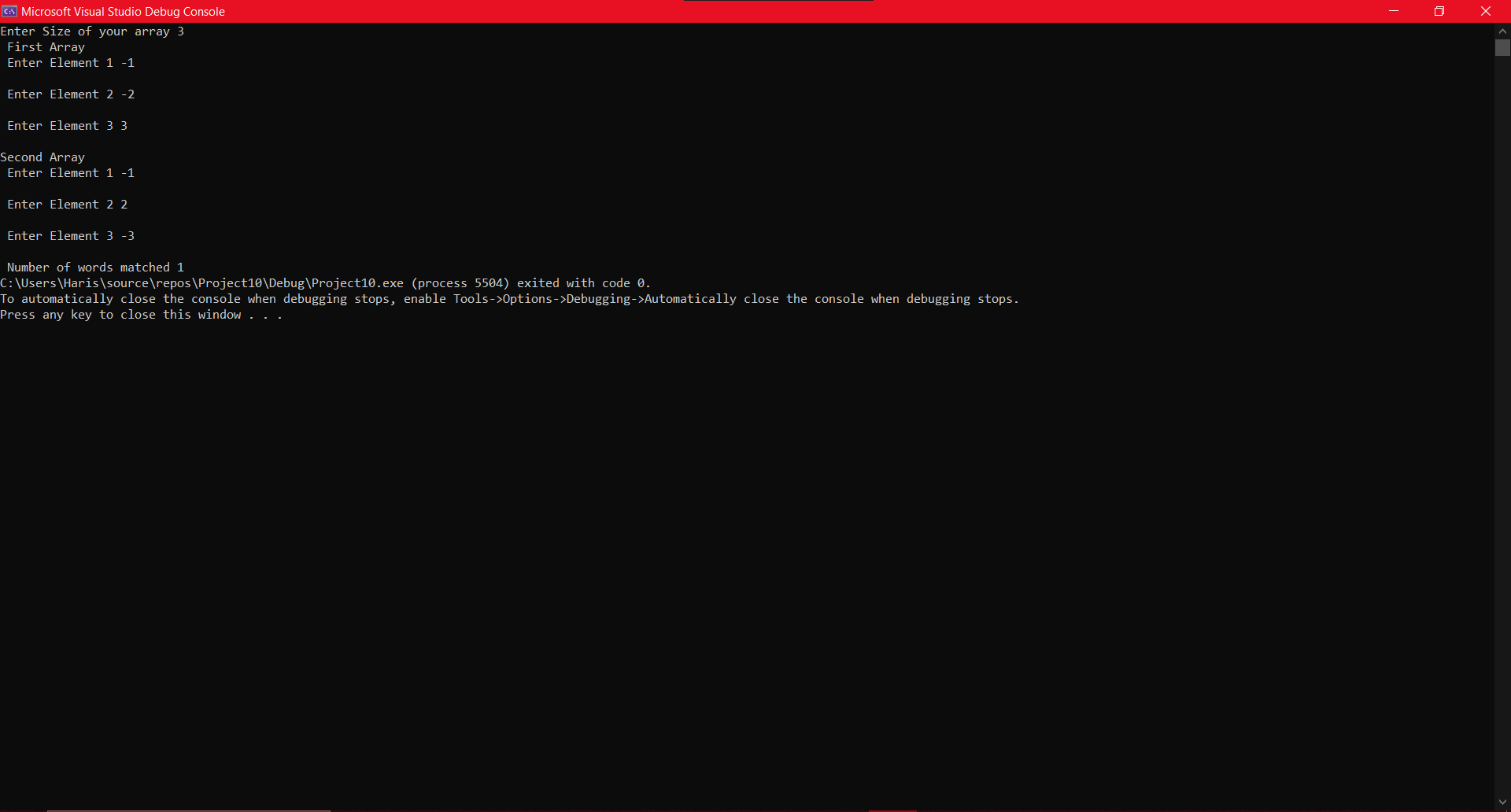
loop l3

ret

print endp

end main

Q1 Output



Q2:

include irvine32.inc

include macros.inc

.data

num qword 9999888812h

num2 qword 6941202021h

array qword ?

extended\_sub proto,hello1:dword,hello12:dword,hello21:dword,hello22:dword

.code

main proc

mov eax,dword ptr num

push eax

mov edx, dword ptr num+4

mov eax,edx

call writehex

pop eax

call writehex

mwrite " Subs "

push eax

mov ebx,dword ptr num2

mov eax,ebx

push eax

mov ecx,dword ptr num2+4

mov eax,ecx

call writehex

pop eax

call writehex

pop eax

invoke extended\_sub, eax,edx,ebx,ecx

mwrite " is equal to "

push eax

mov eax,edx

call writehex

pop eax

call writehex

exit

main endp

extended\_sub proc, hello1:dword,hello12:dword,hello21:dword,num22:dword

mov esi,offset array

mov eax,hello1

mov edx,hello12

mov ebx,hello21

mov ecx,num22

sub eax,ebx

mov [esi],eax

sub edx,ecx

mov [esi+4],edx

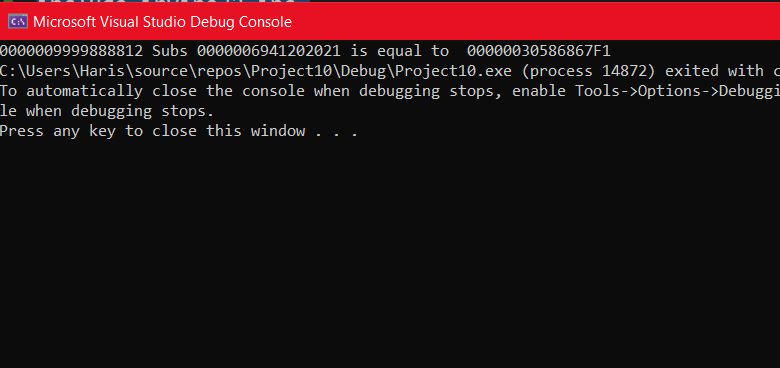
mov eax,dword ptr array

mov edx,dword ptr array+4

ret

extended\_sub endp

end main



**Q3**

Q3

include Irvine32.inc

include macros.inc

LinearSearch PROTO ToSearch : dword

.data

len dword ?

arr sdword 3,324,-524,43,634,237,34,522,1230,123

.code

main PROC

xor eax, eax

xor ebx, ebx

xor ecx, ecx

xor edx, edx

xor esi, esi

xor edi, edi

mwrite "Enter an Element to Search in the array : "

call readint

mov ebx , eax

mov ecx , LENGTHOF arr

mov esi , 0

mov edi , 0

mov eax , 0

INVOKE LinearSearch, ebx

cmp edx , 0

je NotFound

jne exitt

NotFound:

mwrite "Not Found"

exitt:

exit

main endp

LinearSearch PROC ToSearch : dword

cmp ecx , 0

je endd

mov eax , arr[esi]

cmp eax , ToSearch

je Break

add esi , 4

inc edi

dec ecx

Invoke LinearSearch , ToSearch

endd:

ret

Break:

mwrite "Found at Index : "

mov eax , edi

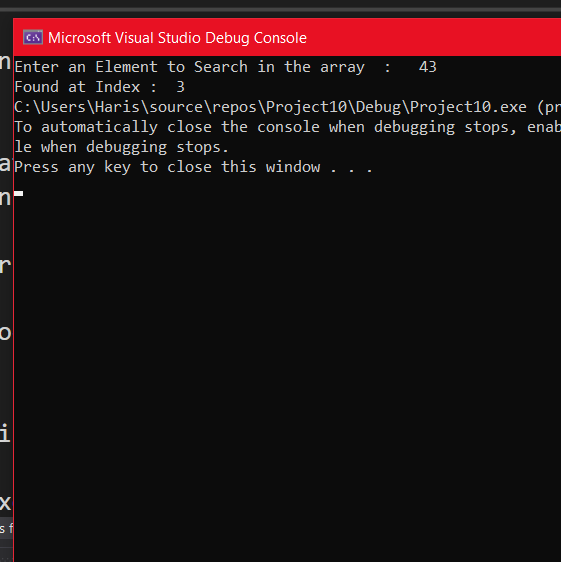
mov edx , 1

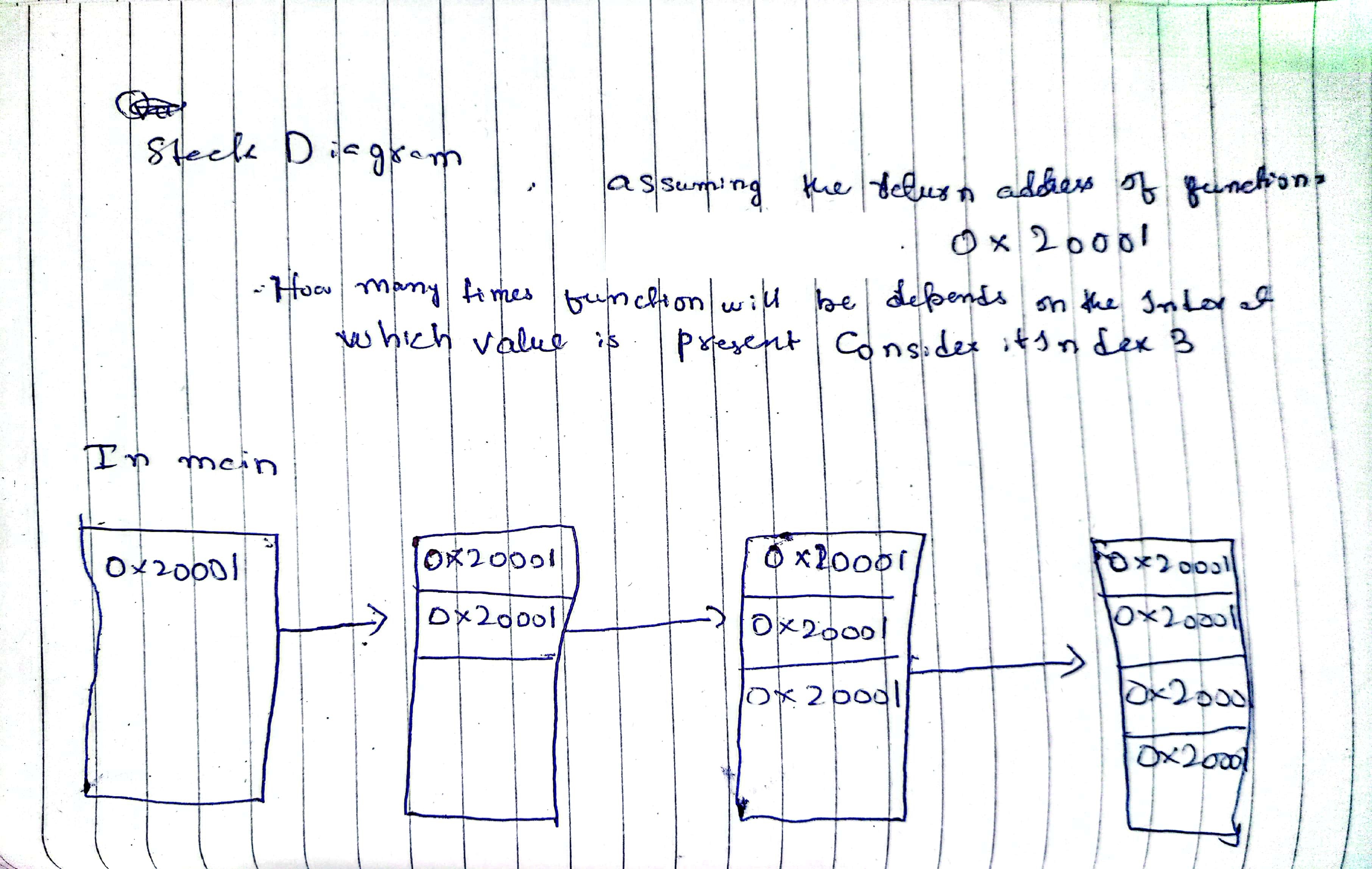
call writedec

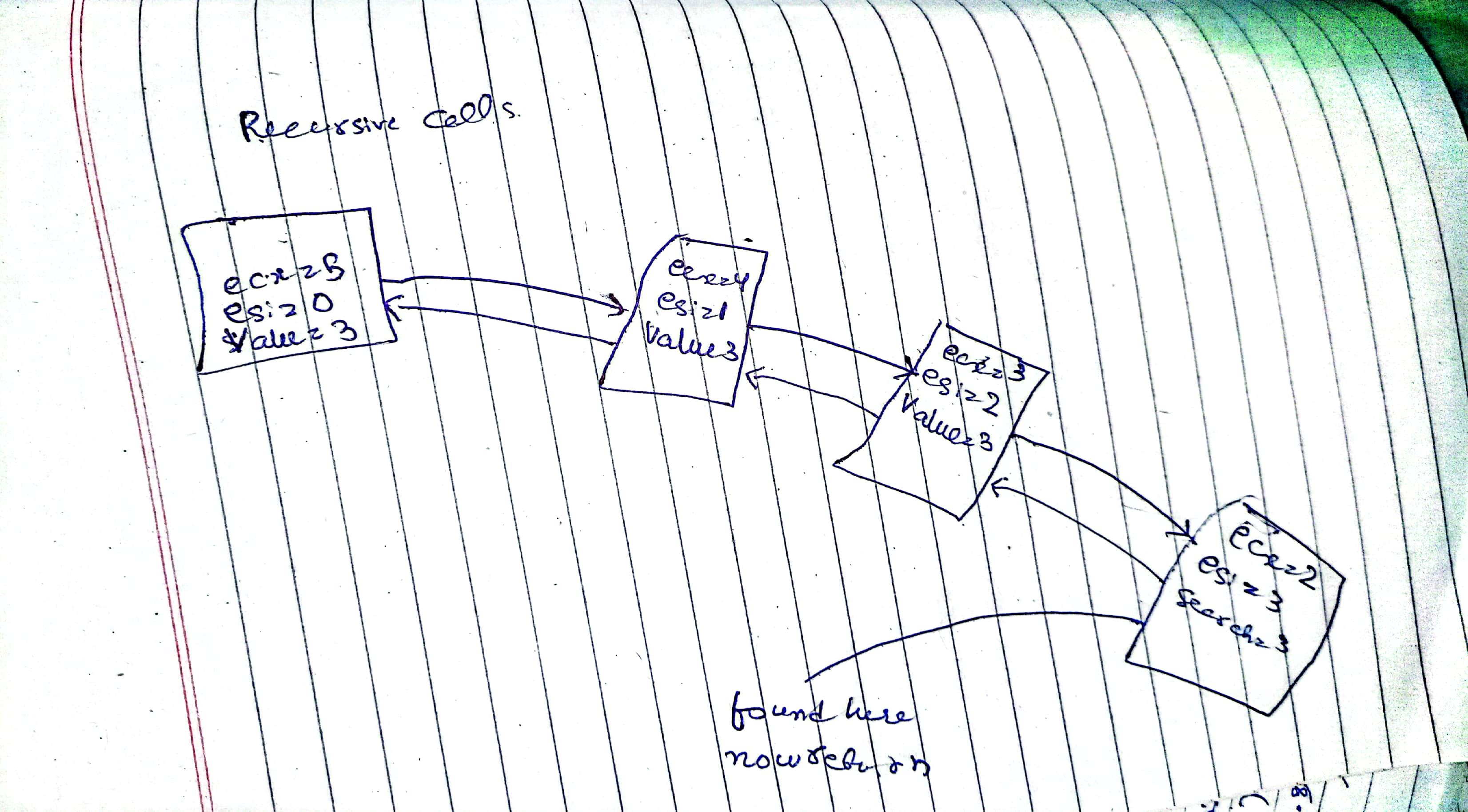
ret

LinearSearch endp

end main







Q4

include irvine32.inc

include macros.inc

.data

array dword 100 dup(?)

len dword ?

count dword 0

.code

FindThrees PROTO,arr:ptr DWORD,s: DWORD

main PROC

; Initialize registers

XOR EAX, EAX

XOR EBX, EBX

XOR ECX, ECX

MWRITE "Enter size of your array: "

CALL ReadInt

MOV LEN, EAX

MOV ECX, LEN

MOV ESI, OFFSET array

L1:

MWRITE "Enter element: "

INC EBX

MOV EAX, EBX

CALL WriteDec

MWRITE " -> "

CALL ReadInt

MOV [ESI], EAX

ADD ESI, 4

CALL CRLF

LOOP L1

INVOKE FindThrees, ADDR array, LEN

call writedec

EXIT

main ENDP

FindThrees PROC, arr: PTR DWORD, s: DWORD

MOV ECX, s

MOV ESI, arr

MOV EAX, [ESI]

ADD ESI, TYPE array

dec ecx

L2:

CMP EAX, [ESI]

JE PLUS

JMP LAST2

PLUS:

INC COUNT

Jmp last1

last2:

mov eax,[esi]

LAST1:

CMP COUNT, 2

JE GO

ADD ESI, TYPE array

LOOP L2

MOV EAX, 0

JMP LAST

GO:

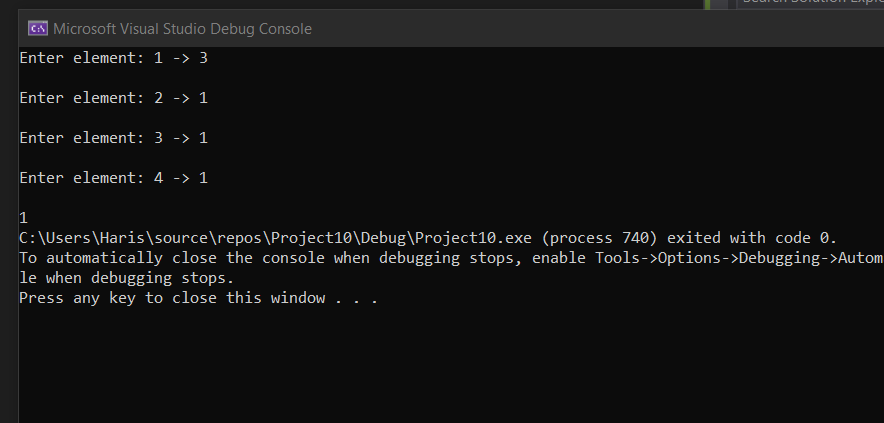
MOV EAX, 1

LAST:

RET

FindThrees ENDP

END main



Q5:

include irvine32.inc

include macros.inc

buffersize=500

.data

str1 byte buffersize dup(?)

len dword ?

len2 dword ?

count dword ?

.code

strtrim proc,arr:ptr byte ,s:dword

mov esi,arr

xor edi,edi

mov edi,0

inc edi

mov eax,s

l:

xor edx,edx

mov dl, str1[edi]

cmp edx,eax

jne swap

inc edi

jmp last

swap:

xor ebx,ebx

mov bl,str1[edi]

mov [esi],bl

mov str1[edi],al

inc esi

inc count

last:

cmp edi,len

je go

jmp l

go:

mov eax,count

mov str1[eax],0

ret

strtrim endp

main PROC

XOR EAX, EAX

XOR EBX, EBX

XOR ECX, ECX

xor edx,edx

xor esi,esi

xor edi,edx

xor eax,eax

mov edx,offset str1

mov ecx,lengthof buffersize

call readstring

mov len,eax

mov str1[eax],0

mwrite "Original string "

call crlf

mov edx,offset str1

call writestring

call crlf

movzx eax,str1[0]

call crlf

invoke strtrim,addr str1,eax

mwrite "Trimmed String -> "

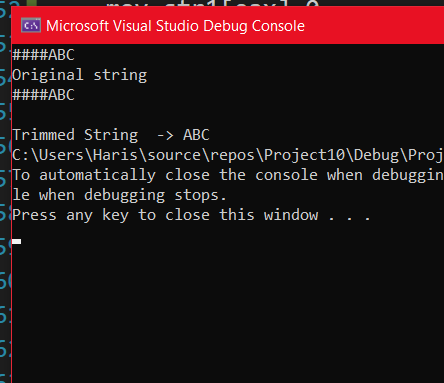
mov edx,offset str1

call writestring

exit

main endp

end main



**Q6**

include irvine32.inc

include macros.inc

.data

.code

divideuntil proto,divisor:dword,divident:dword

divideuntil proc,divisor:dword,dividend:dword

mov eax,dividend

mov ebx,divisor

cmp eax,5h

jle go

xor edx,edx

div ebx

invoke divideuntil, ebx,eax

go:

ret

divideuntil endp

main PROC

XOR EAX, EAX

XOR EBX, EBX

XOR ECX, ECX

xor edx,edx

xor esi,esi

xor edi,edx

xor eax,eax

mov eax,0D4A4h

mov ebx,0Ah

invoke divideuntil,ebx,eax

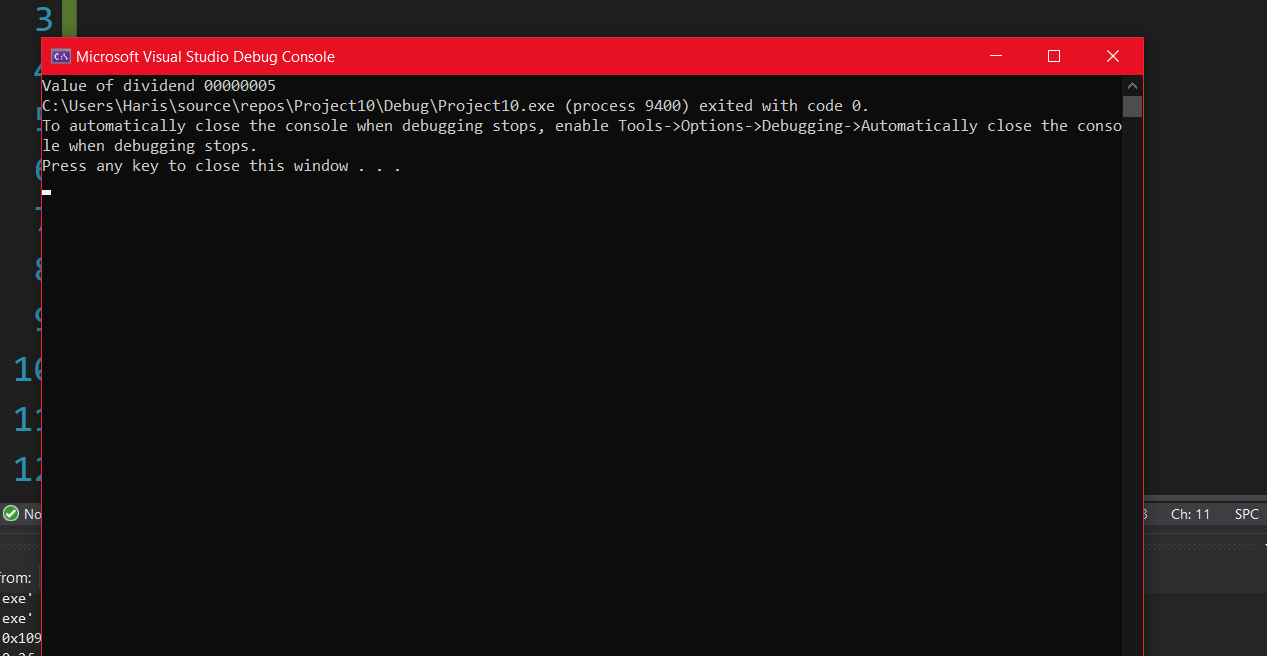
mwrite "Value of dividend "

call writehex

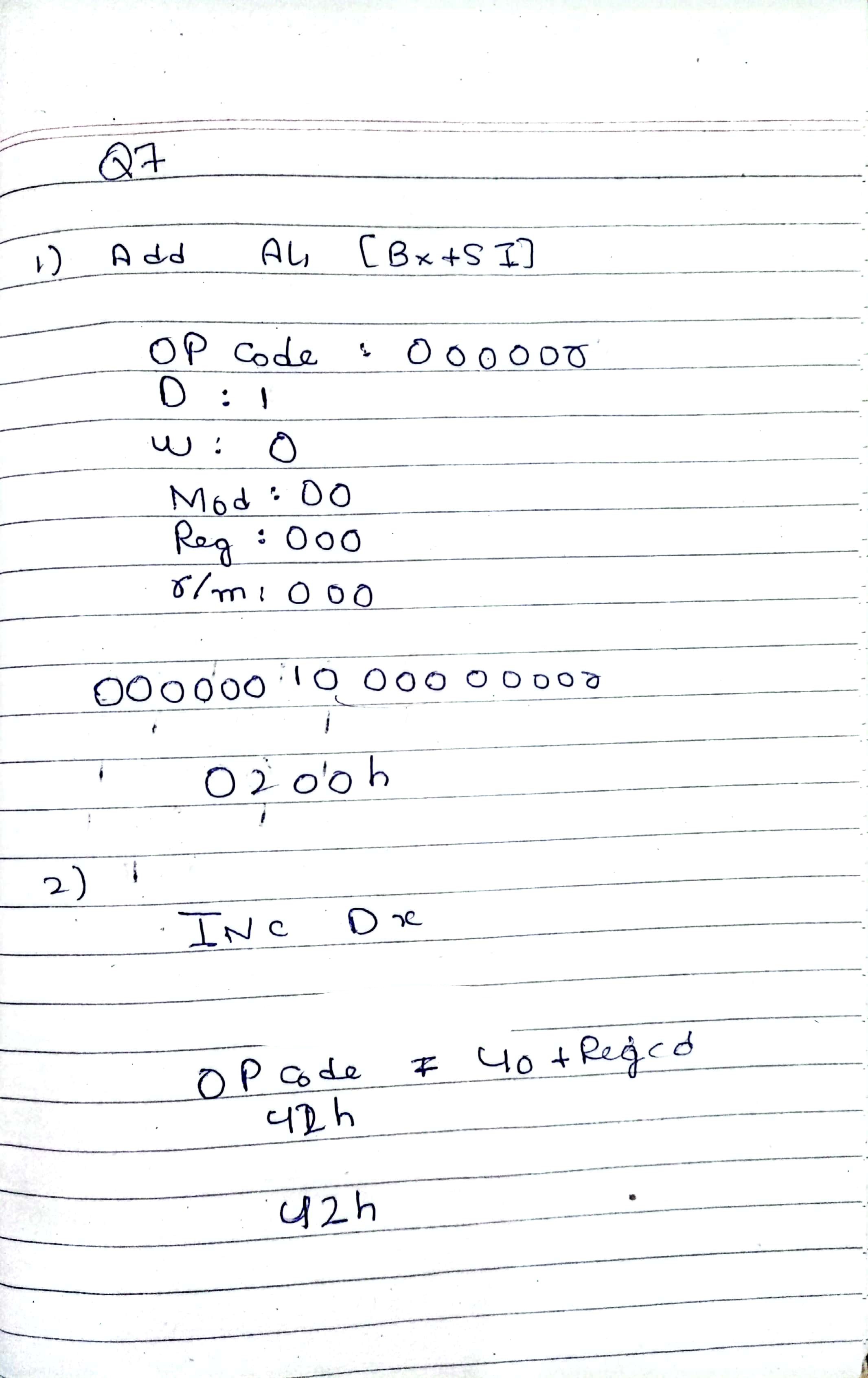
exit

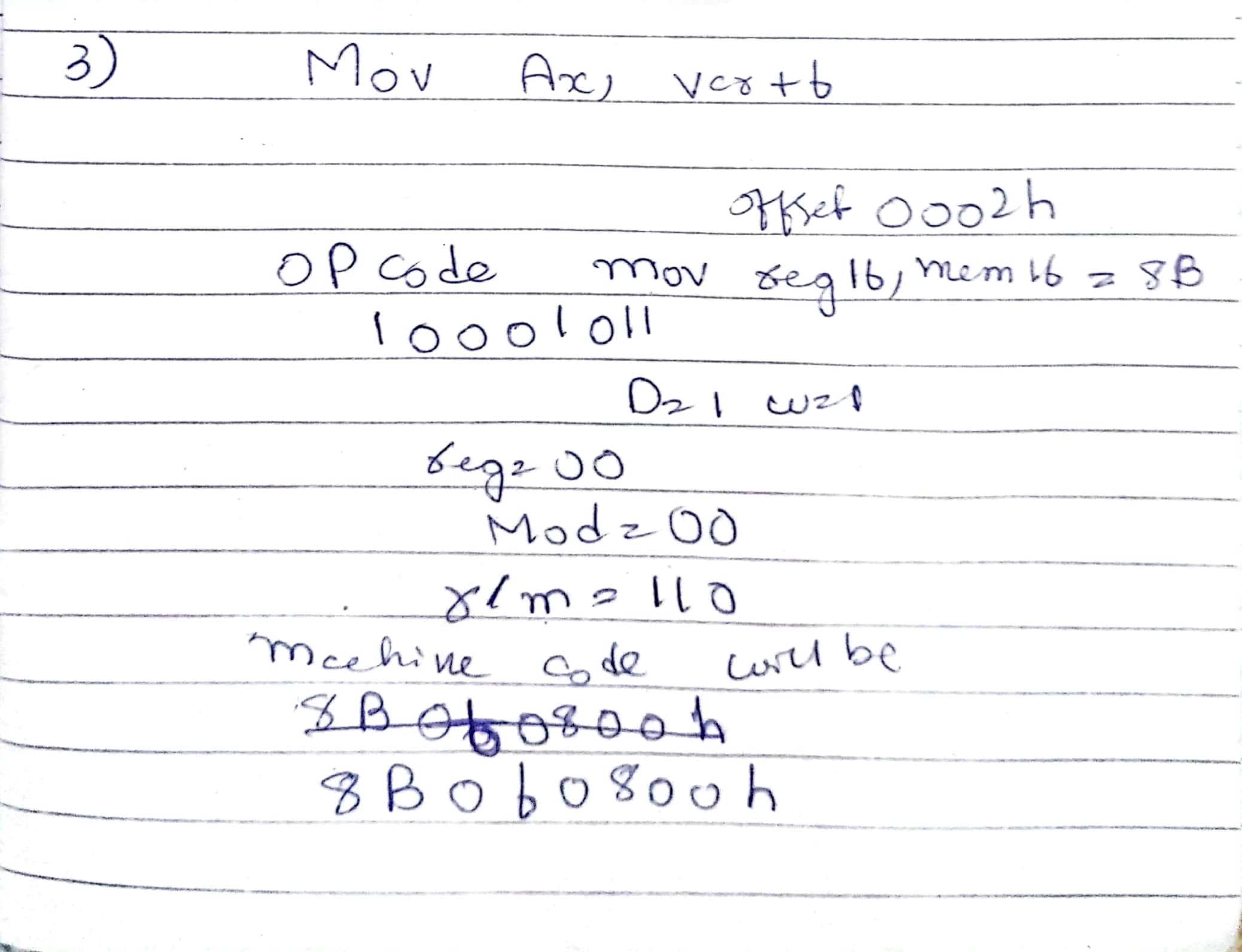
main endp

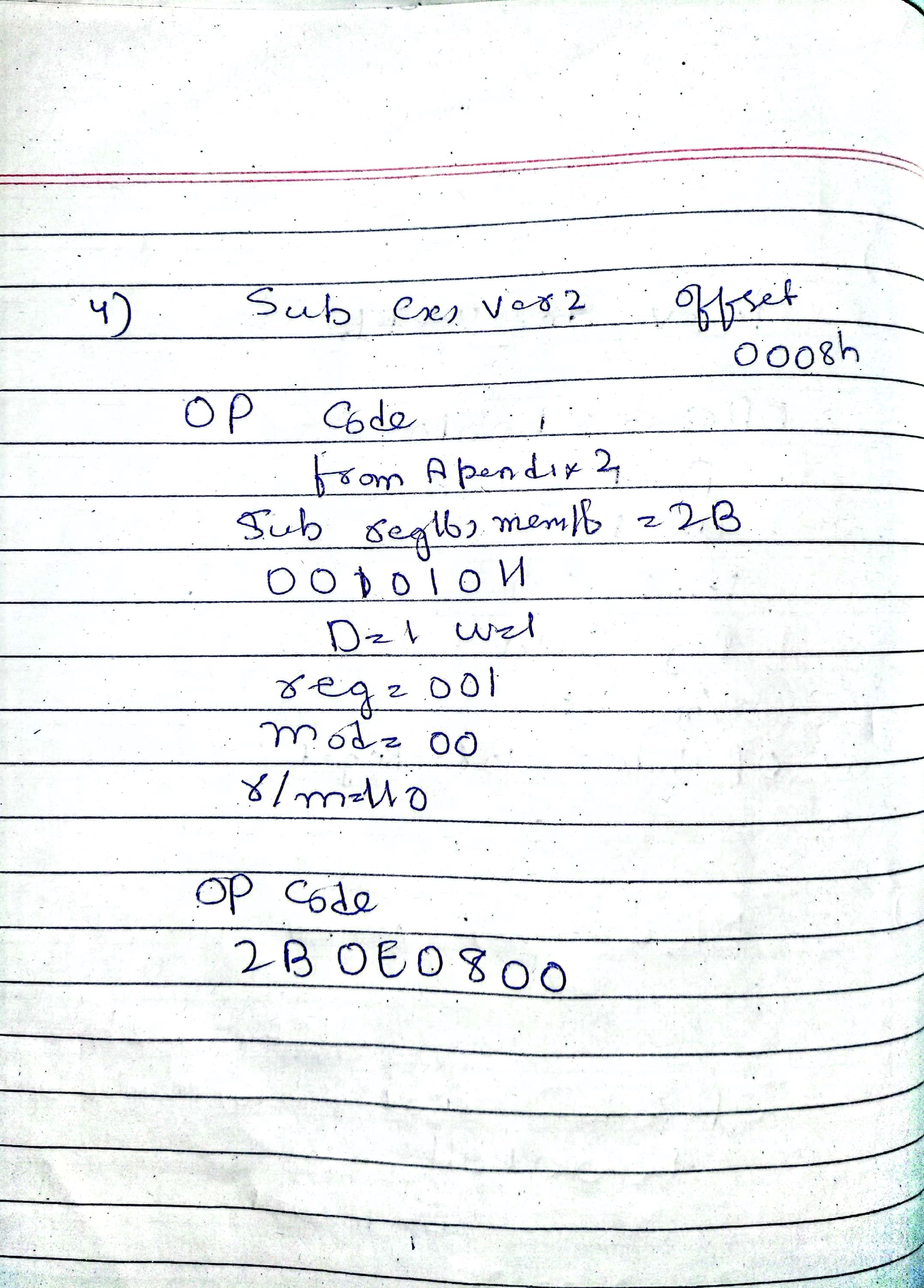
END main

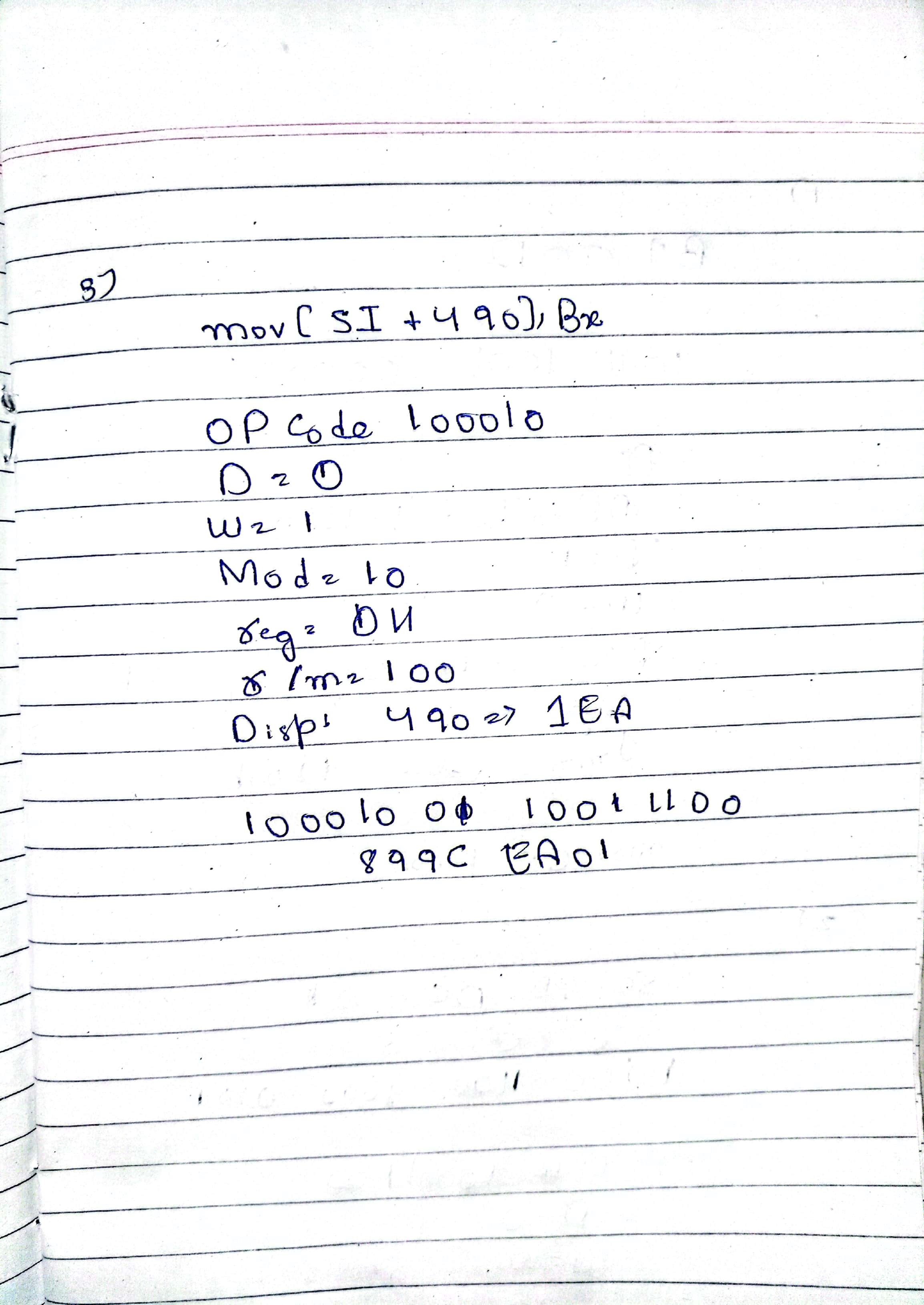


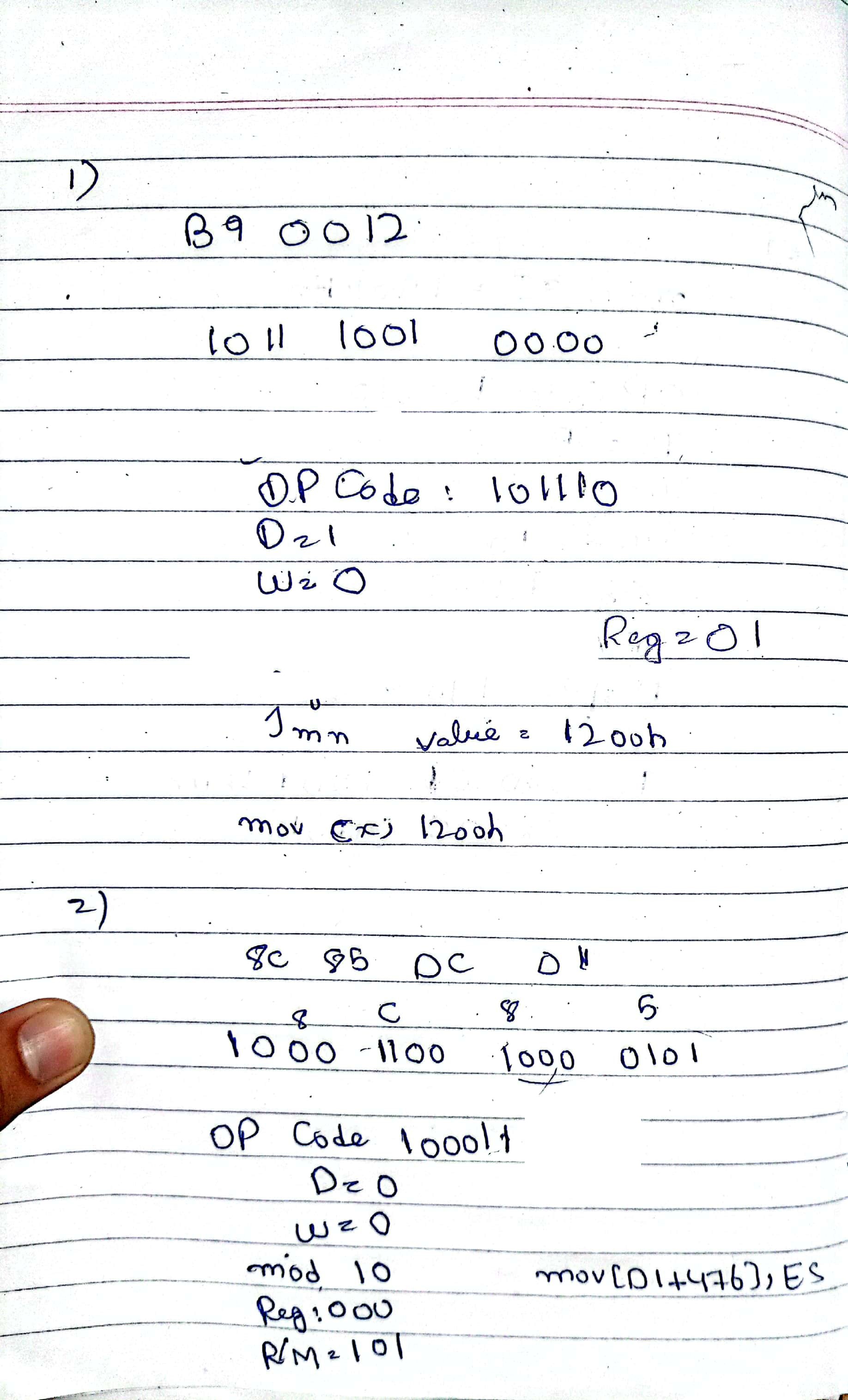
Q7:





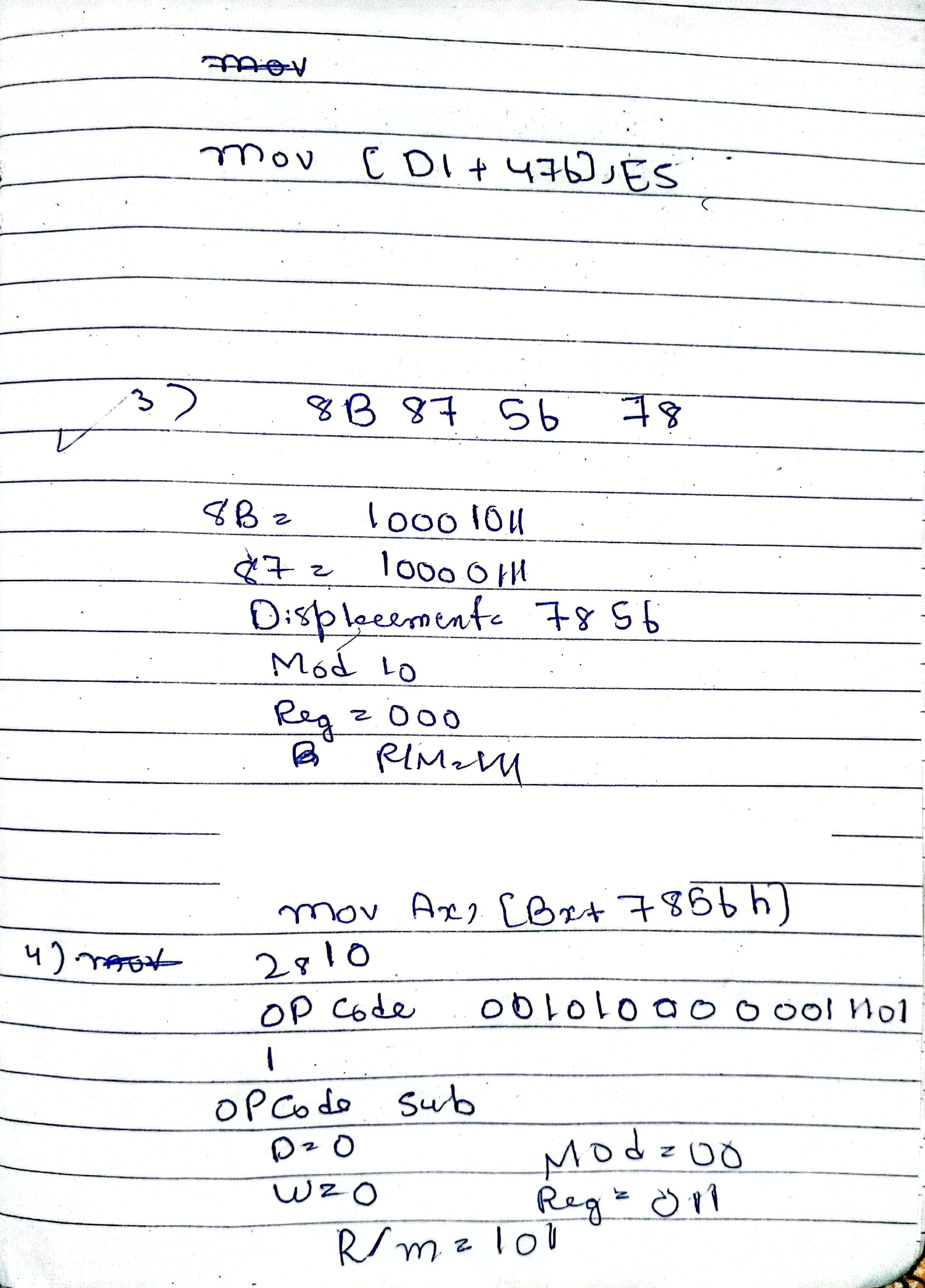


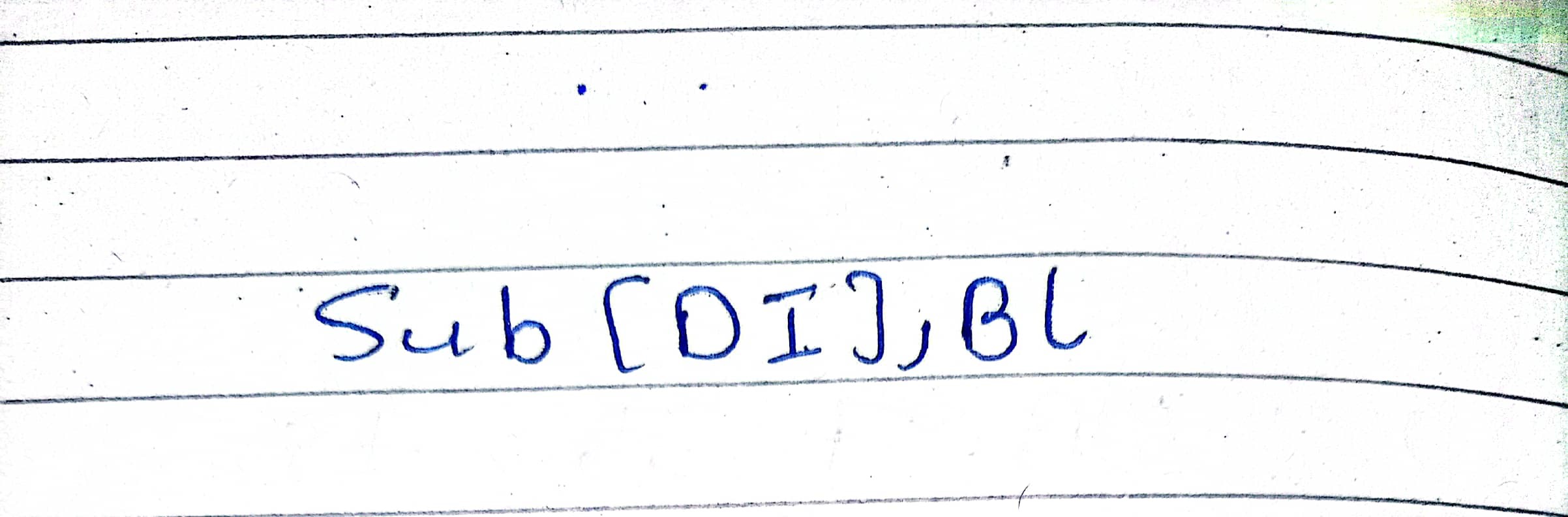




c

Q8





Q9 A)Code

include irvine32.inc

include macros.inc

.data

A word ?

.code

main PROC

XOR EAX, EAX

XOR EBX, EBX

XOR ECX, ECX

xor edx,edx

xor esi,esi

xor edi,edx

xor eax,eax

mwrite "Enter Value OF A "

call readint

mov A,ax

imul ax ,5

sub ax,7

mov A,ax

mov ax,A

call crlf

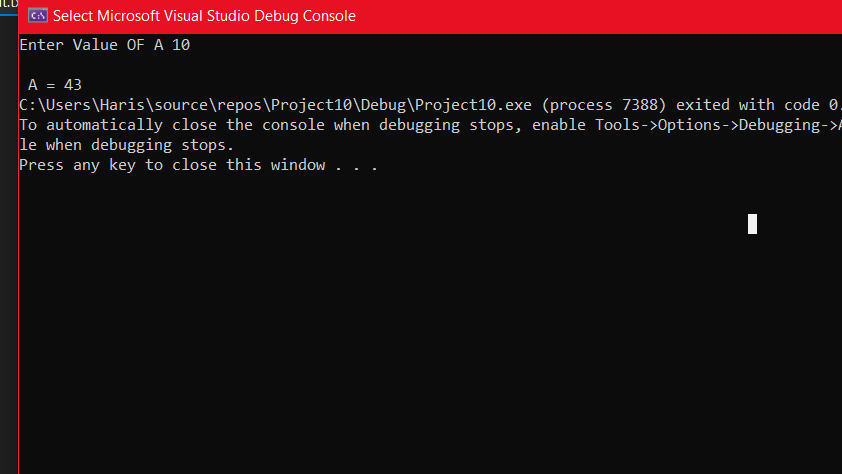
mwrite " A = "

call writedec

exit

main endp

END main



B)

include irvine32.inc

include macros.inc

.data

A word ?

B Word ?

.code

main PROC

XOR EAX, EAX

XOR EBX, EBX

XOR ECX, ECX

xor edx,edx

xor esi,esi

xor edi,edx

xor eax,eax

mwrite "Enter Value OF A "

call readint

mov A,ax

call crlf

xor eax,eax

mwrite "Enter Value OF B "

call readint

mov B,ax

;B=(A-B)\*(B+10)

mov ax,A

sub ax,b

add B,10

imul ax,B

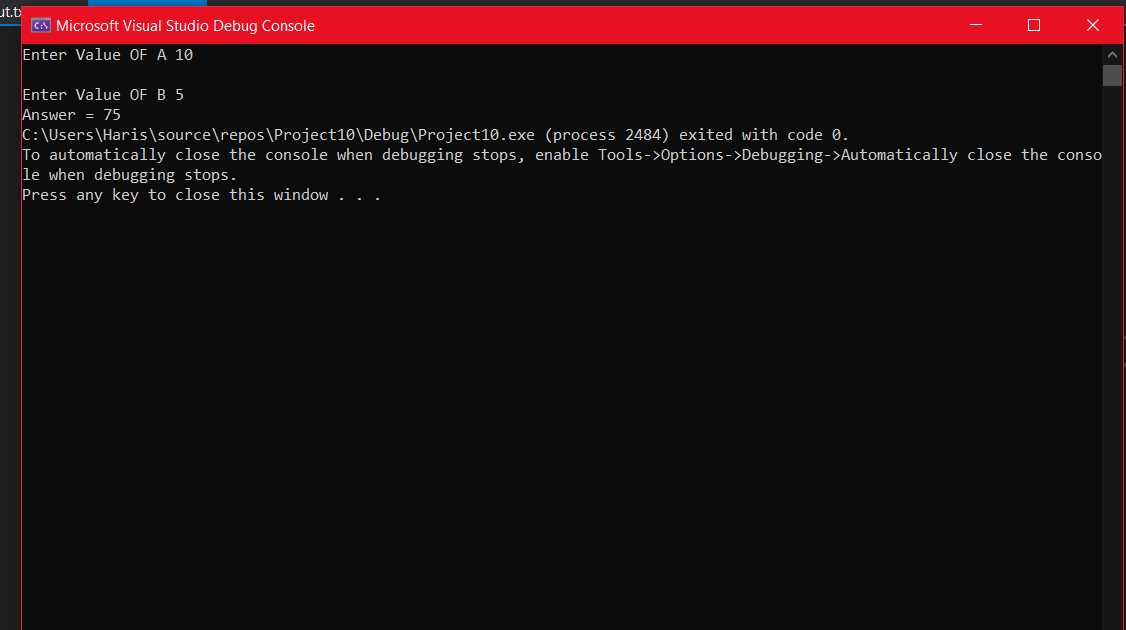
mwrite "Answer = "

call writedec

exit

main endp

END main



C)

include irvine32.inc

include macros.inc

.data

A word ?

B Word ?

.code

main PROC

XOR EAX, EAX

XOR EBX, EBX

XOR ECX, ECX

xor edx,edx

xor esi,esi

xor edi,edx

xor eax,eax

mwrite "Enter Value OF A "

call readint

mov A,ax

call crlf

;A=6-9\*A

imul ax,9

mov bx,6

sub bx,ax

movsx eax, bx

mov A,ax

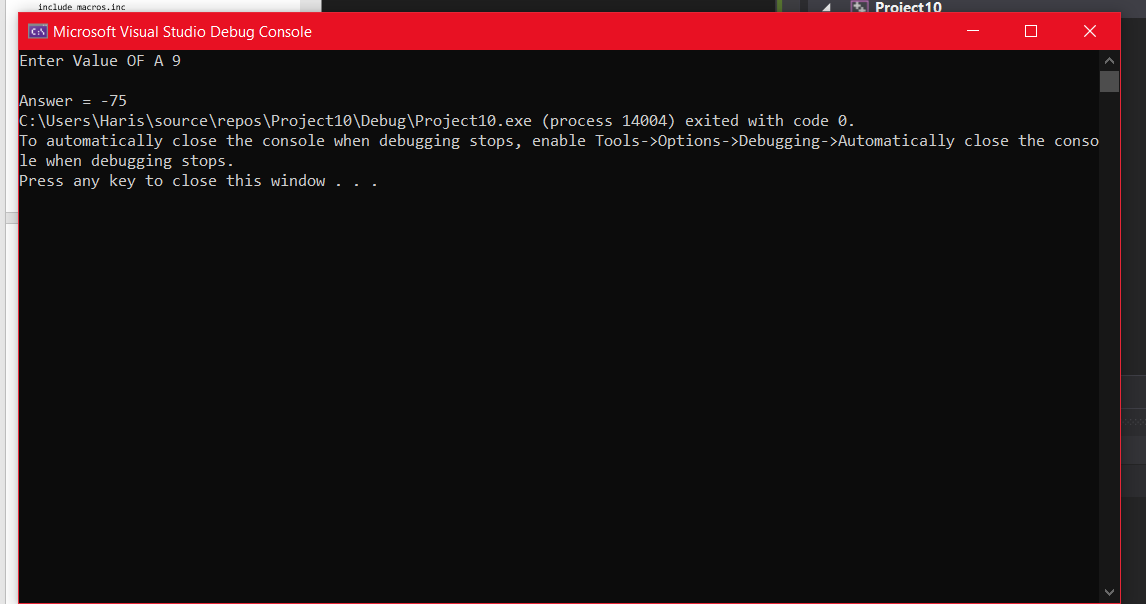
mwrite "Answer = "

call writeint

exit

main endp

END main



D)

include irvine32.inc

include macros.inc

.data

A word ?

B Word ?

C1 word ?

.code

main PROC

XOR EAX, EAX

XOR EBX, EBX

XOR ECX, ECX

xor edx,edx

xor esi,esi

xor edi,edx

xor eax,eax

mwrite "Enter Value OF A "

call readint

mov A,ax

call crlf

mwrite "Enter Value OF B "

call readint

mov B,ax

call crlf

mwrite "Enter Value OF C "

call readint

mov C1,ax

call crlf

mov ax,a

mov bx,b

mov dx,c1

imul ax,ax

imul bx,bx

imul dx,dx

add ax,bx

cmp dx,ax

je go

clc

jmp last

go:

stc

last:

call dumpregs

exit

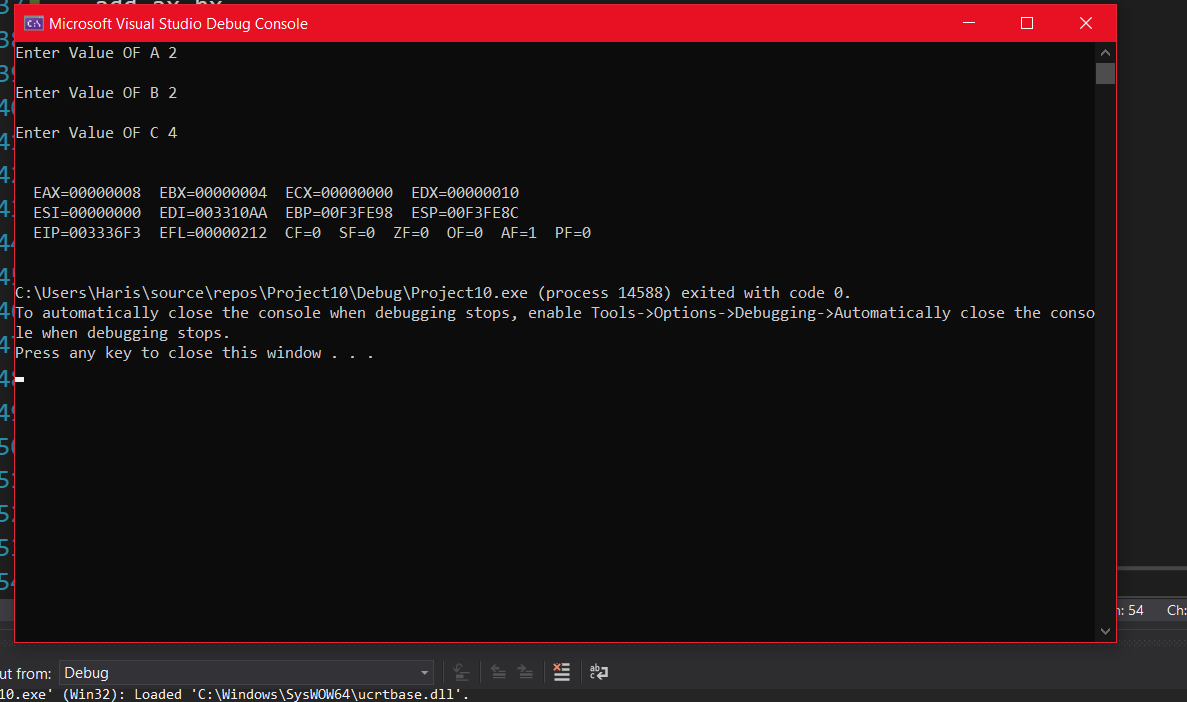
main endp

END main

Output 25=(3)^2+(4)^2 25=9+16



Output2 of Q9 D)



**Q10:**

include irvine32.inc

include macros.inc

.data

M dword ?

N Dword ?

.code

main PROC

XOR EAX, EAX

XOR EBX, EBX

XOR ECX, ECX

xor edx,edx

xor esi,esi

xor edi,edx

xor eax,eax

mwrite "Enter Two Numbers to find their gcd "

call crlf

mwrite "Enter First Number -> "

call readint

mov M,eax

mwrite "Enter Second Number -> "

call readint

mov n,eax

cmp M,eax

jg go

mov eax,N ; for storing greater value in eax

mov ebx,M

mov M,eax

mov N,ebx

jmp l1

go:

mov eax,M

l1:

mov edx, 0

div N

mov eax,N

cmp edx,0

je quit

mov N,edx

jmp l1

quit:

mov eax, N

call crlf

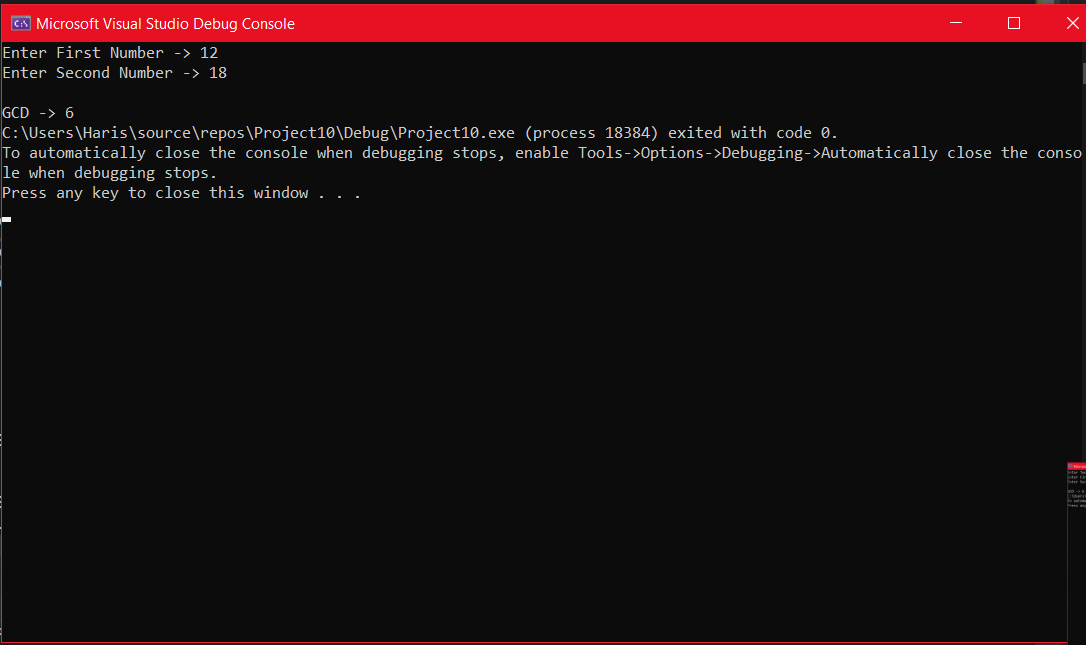
mwrite "GCD -> "

call writedec

exit

main endp

end main



**Q11 Code**

include irvine32.inc

include macros.inc

.data

array dword 67,45,98,33

dword 70,56,87,44

dword 82,72,89,40

dword 80,67,95,50

dword 78,76,92,60

i dword 0

j dword 5

.code

main PROC

XOR EAX, EAX

XOR EBX, EBX

XOR ECX, ECX

xor edx,edx

xor esi,esi

xor edi,edx

xor eax,eax

mov ecx,4

l1:

push ecx

mov ecx,5

xor eax,eax

l2:

add eax,array[ebx+lengthof array \* edx]

add edx,4

loop l2

pop ecx

add ebx,4

mov edx,0

div j

mwrite " Average is "

call writedec

call crlf

loop l1

call dumpregs

xor ebx,ebx

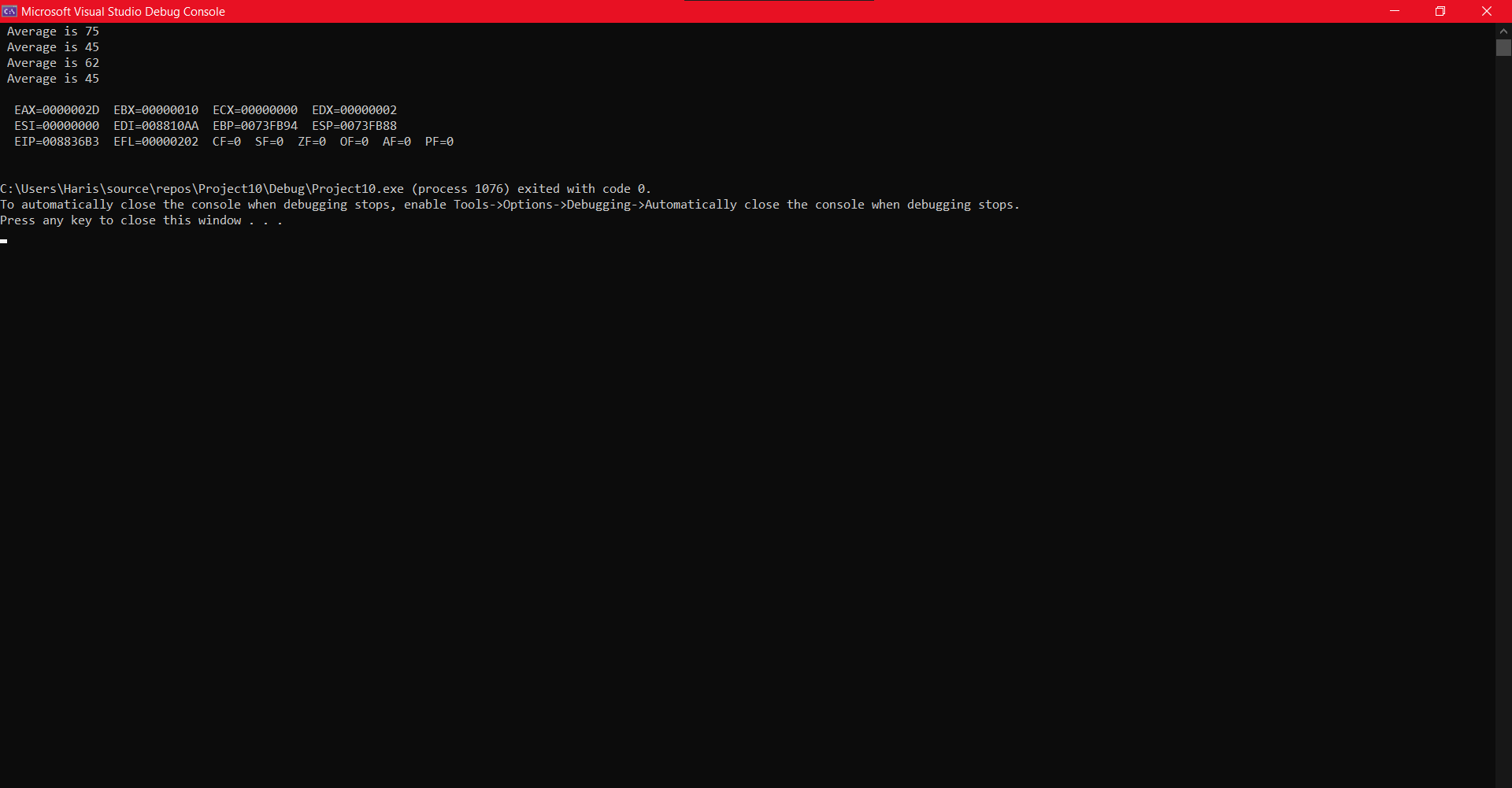
mov ebx,5

div ebx

exit

main endp

END main



Q12: Note First create five files .txt store content in it then run this program for feasibility I am attaching five documents

Include irvine32.inc

Include macros.inc

buffersize=1000

.data

string BYTE buffersize dup(?)

testString1 BYTE buffersize dup(?)

testString2 BYTE buffersize dup(?)

testString3 BYTE buffersize dup(?)

testString4 BYTE buffersize dup(?)

testString5 BYTE buffersize dup(?)

len1 dword ?

len2 dword ?

len3 dword ?

len4 dword ?

len5 dword ?

File2 Handle ?

File3 Handle ?

File4 Handle ?

File5 Handle ?

File6 Handle ?

wordMatch dword 0

doc1 byte "Doc1.txt",0

doc2 byte "Doc2.txt",0

doc3 byte "Doc3.txt",0

doc4 byte "Doc4.txt",0

doc5 byte "Doc5.txt",0

foundMsg BYTE "Plagiarism detected in test document ",0

notFoundMsg BYTE "No plagiarism is detected in test document ",0

.code

checkConsecutiveWords PROC USES ecx ebx

checkNext:

mov al, [esi+ecx]

cmp al, 0

je endstring

mov al, [edi+ebx]

cmp al, 0

je endstring

cmp al, "."

je wordCompleted

mov al, [esi+ecx]

cmp al, "."

je wordCompleted

cmp al, [edi+ebx]

jne letterNotEqual

cmp al, " "

jne notWordComplete

wordCompleted:

inc wordMatch

mov edx, wordMatch

cmp wordMatch, 3

jge foundPlag

notWordComplete:

inc ebx

inc ecx

jmp checkNext

endstring:

cmp wordMatch, 2

jge foundPlag

letterNotEqual:

mov eax, 0

ret

foundPlag:

mov eax, 1

ret

checkCOnsecutiveWords ENDP

checkString PROC USES ecx string1: PTR BYTE, string2: PTR BYTE, size1: DWORD, size2: DWORD

mov esi, string1

mov edi, string2

mov ecx, 0

mov ebx, 0

whileLoop1:

call checkConsecutiveWords

cmp eax, 1

je foundPlag

mov wordMatch, 0

while2:

inc ecx

mov eax, size1

cmp ecx, eax

jge moveToNextWord

mov al, [esi+ecx]

cmp al, " "

jne while2

inc ecx

jmp whileLoop1

moveToNextWord:

while3:

inc ebx

mov eax, size2

cmp ebx, eax

jge break

mov al, [edi+ebx]

cmp al, " "

jne while3

inc ebx

mov ecx, 0

jmp whileLoop1

foundPlag:

mov eax, 1

ret

break:

mov eax, 0

ret

checkString ENDP

PrintMsg PROC

cmp eax, 1

je foundPlag

mov edx, OFFSET notFoundMsg

call writeString

mov eax, ecx

call writeDec

ret

foundPlag:

mov edx, OFFSET foundMsg

call writeString

mov eax, ecx

call writeDec

ret

PrintMsg ENDP

main PROC

mwrite "Enter COntent Of Text Document "

mov ecx,buffersize

mov edx,offset string

call readstring

mov len1,eax

mov string[eax],0

mov edx,offset doc1

call openinputfile

mov file2,eax

mov edx,offset teststring1

mov ecx,sizeof teststring1

call readfromfile

xor edx,edx

xor ecx,ecx

xor eax,eax

mov edx,offset doc2

call openinputfile

mov file3,eax

mov edx,offset teststring2

mov ecx,sizeof teststring2

call readfromfile

xor edx,edx

xor ecx,ecx

xor eax,eax

mov edx,offset doc3

call openinputfile

mov file4,eax

mov edx,offset teststring3

mov ecx,sizeof teststring3

call readfromfile

xor edx,edx

xor ecx,ecx

xor eax,eax

mov edx,offset doc4

call openinputfile

mov file5,eax

mov edx,offset teststring4

mov ecx,sizeof teststring4

call readfromfile

xor edx,edx

xor ecx,ecx

xor eax,eax

mov edx,offset doc5

call openinputfile

mov file6,eax

mov edx,offset teststring5

mov ecx,sizeof teststring5

call readfromfile

xor eax,eax

xor ecx,ecx

xor edx,edx

mov ecx, 1

INVOKE checkString, ADDR testString1, ADDR string, LENGTHOF testString1, LENGTHOF string

call PrintMsg

call Crlf

mov wordMatch, 0

inc ecx

INVOKE checkString, ADDR testString2, ADDR string, LENGTHOF testString2, LENGTHOF string

call PrintMsg

call Crlf

mov wordMatch, 0

inc ecx

INVOKE checkString, ADDR testString3, ADDR string, LENGTHOF testString3, LENGTHOF string

call PrintMsg

call Crlf

mov wordMatch, 0

inc ecx

INVOKE checkString, ADDR testString4, ADDR string, LENGTHOF testString4, LENGTHOF string

call PrintMsg

call Crlf

mov wordMatch, 0

inc ecx

INVOKE checkString, ADDR testString5, ADDR string, LENGTHOF testString5, LENGTHOF string

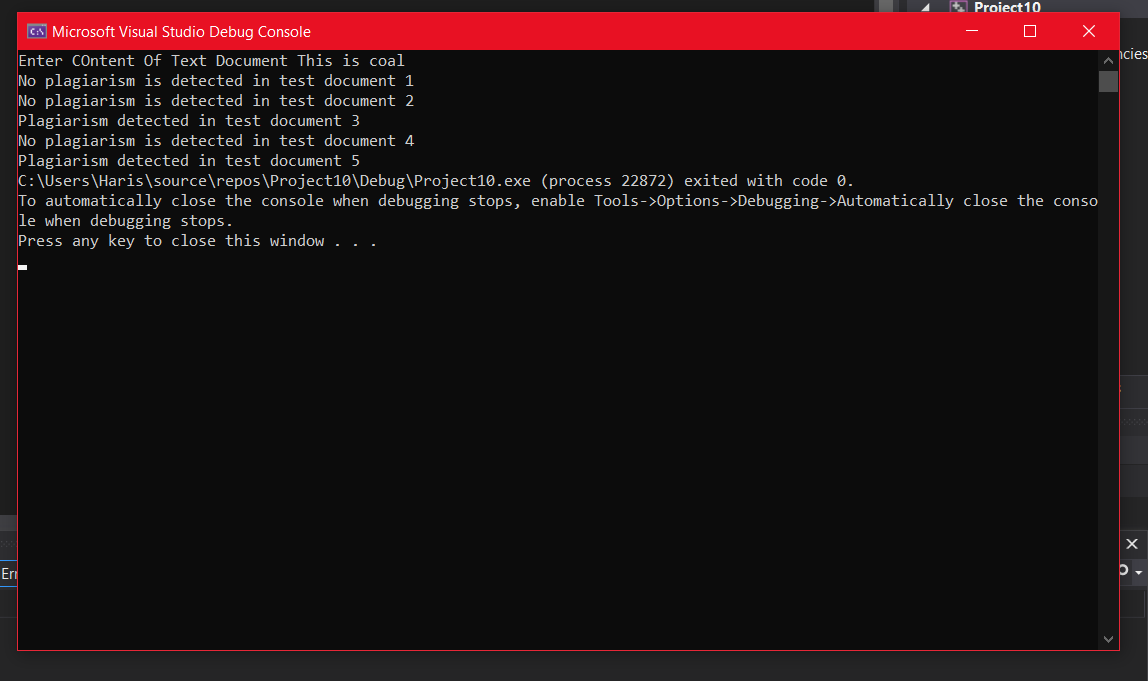
call PrintMsg

mov wordMatch, 0

exit

main ENDP

end main





DOC 1 Contains

This is My Document

Doc 2 Contains

This is Hers Document

DOC 3 contains

This is coal project

Doc 4 Contains

This is yours document

Doc 5 Contains

This is coal assignment