National University of Computer and Emerging Sciences

COAL Lab Midterm

Computer Organization and Assembly Language

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time Allowed** | 90 Minutes | **Student Name** | |  |
| **Maximum Marks** | 100 | **Roll Number** | |  |
| **Lab Instructors** | Hazoor | | **Date** |  |

**Before you start make sure:**

1. **Fill word Format on GCR & submit PDF [Only PDF formats accepted]**
2. **PLAGIARISM WILL BE MARKED ZERO WITH NO RETAKE**

# **Activity 1: [20 Marks]**

Initialize a memory array with last 4 digits of **Your Own Roll Number** (for example, if your roll number is **16L-4195** then memory array should be initialized with **{4,1,9,5}**).

Then write a subroutine **LoadMN** which stores the matrices and using the values as described in the Code section below:

# **Activity 2: [50 Marks]**

Use the subroutine from **Activity 1**, to initialize matrices **M** and **N.**

**Theory**:

If and are two matrices of order as shown below

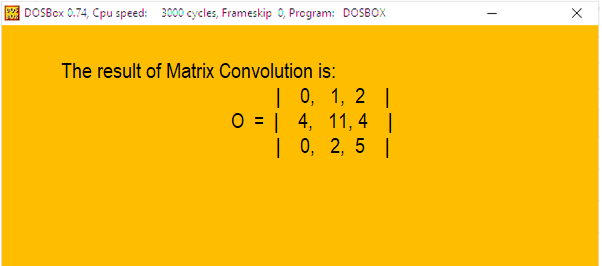
**Convolution** of two matrices can be calculated by padding first matrix and scrolling the second matrix on it across the rows and columns of A and performing dot product:

As shown above, convolution of M and N is the matrix O.

Write a subroutine, **MatConv,** which performs 2x2 matrix **convolution** using the values of M and N initialized in the **Activity 1.**

# **Activity 3: [30 Marks]**

Make a subroutine for printing the output matrix from **Activity 2** on the console as shown below



# **Code:**

**Your final program structure should look like:**

[org 0x0100]

jmp start

rollnum: db 5,1,5,7

a: db 0

b: db 0

c1: db 0

d: db 0

M1: db 0,0

M2: db 0,0

N1: db 0,0

N2: db 0,0

Mr1: db 0,0,0,0

Mr2: db 0,0,0,0

Mr3: db 0,0,0,0

Mr4: db 0,0,0,0

Or1: dw 0,0,0

Or2: dw 0,0,0

Or3: dw 0,0,0

matrix: dw 'O ='

matrixsize: dw 3

message: dw 'The result of Matrix Convolution is:'

messagesize: dw 36

clearscreen:

mov ax, 0xb800

mov es, ax

mov di, 0

nextchar:

mov word [es:di], 0x0720

add di, 2

cmp di, 4000

jne nextchar

ret

LoadMN:

mov si,0

mov bl ,[rollnum+si]

mov [a],bl

inc si

mov bl ,[rollnum+si]

mov [b],bl

inc si

mov bl ,[rollnum+si]

mov [c1],bl

inc si

mov bl ,[rollnum+si]

mov [d],bl

mov si,0

mov di,1

mov al,[a]

mov [M1+si],al

mov [N2+di],al

inc si

dec di

mov al,[b]

mov [M1+si],al

mov [N2+di],al

mov si,0

mov di,1

mov al,[c1]

mov [M2+si],al

mov [N1+di],al

inc si

dec di

mov al,[d]

mov [M2+si],al

mov [N1+di],al

ret

printnum :

push bp

mov bp, sp

push es

push ax

push bx

push cx

push dx

push di

mov ax, 0xb800

mov es, ax

mov ax, [bp + 4]

mov bx, 10

mov cx, 0

nextdigit:

mov dx, 0

div bx

add dl, 0x30

push dx

inc cx

cmp ax, 0

jnz nextdigit

nextpos:

pop dx

mov dh, 0x07

mov[es:di], dx

add di, 2

loop nextpos

pop di

pop dx

pop cx

pop bx

pop ax

pop es

pop bp

ret 2

loadMr:

push si

push di

push ax

mov si,1

mov al,[a]

mov [Mr2 + si],al

add si,1

mov al,[b]

mov [Mr2 + si],al

mov si,1

mov al,[c1]

mov [Mr3 + si],al

add si,1

mov al,[d]

mov [Mr3 + si],al

pop ax

pop di

pop si

ret

orderMatrix:

mov al,[a]

mov bl,al

mul bl

mov si,0

mov [Or1 + si],ax

add si,2

mov al,[a]

mov bl,[b]

mul bl

shl ax,1

mov [Or1+si],ax

add si,2

mov al,[b]

mov bl,[b]

mul bl

mov [Or1+si],ax

mov si,0

mov al,[c1]

mov bl,[a]

mul bl

shl ax,1

mov [Or2+si],ax

add si,2

mov al,[a]

mov bl,[d]

mul bl

mov dx,ax

mov al,[b]

mov bl,[c1]

mul bl

add ax,dx

shl ax,1

mov [Or2+si],ax

add si,2

mov al,[b]

mov bl,[d]

mul bl

shl ax,1

mov [Or2+si],ax

mov si,0

mov al,[c1]

mov bl,[c1]

mul bl

mov [Or3+si],ax

add si,2

mov al,[c1]

mov bl,[d]

mul bl

shl ax,1

mov [Or3+si],ax

add si,2

mov al,[d]

mov bl,[d]

mul bl

mov [Or3+si],ax

ret

ptrstr:

push bp

mov bp,sp

push es

push ax

push cx

push si

push di

mov ax,0xb800

mov es,ax

mov si,[bp+6]

mov cx,[bp+4]

mov ah,0x07

nextchar1:

mov al,[si]

mov [es:di],ax

add di,2

add si,1

loop nextchar1

pop di

pop si

pop cx

pop ax

pop es

pop bp

ret 4

start:

call clearscreen

call LoadMN

call loadMr

mov di,160

mov ax,message

push ax

push word[messagesize]

call ptrstr

call orderMatrix

mov bx,Or1

mov si,0

mov di,330

mov cx,3

mov ax,0

mov word[es:di],0x7020

add di,4

l3:

mov ax, [bx + si]

push ax

call printnum

add di, 10

add si, 2

loop l3

sub di,4

mov word[es:di],0x7020

mov bx,Or2

mov si,0

mov di,490

mov cx,3

mov ax,0

mov word[es:di],0x7020

add di,4

l4:

mov ax, [bx + si]

push ax

call printnum

add di, 10

add si, 2

loop l4

sub di,4

mov word[es:di],0x7020

mov di,480

mov ax,matrix

push ax

push word[matrixsize]

call ptrstr

mov bx,Or3

mov si,0

mov di,650

mov cx,3

mov ax,0

mov word[es:di],0x7020

add di,4

l5:

mov ax, [bx + si]

push ax

call printnum

add di, 10

add si, 2

loop l5

sub di,4

mov word[es:di],0x7020

mov ax,0x4c00

int 0x21