

# **CS1107:**

# **COMPUTER ARCHITECTURE**

## **LAB FILE**

## **CODES**

**SUBMITTED BY:**  
KONDROLLA DINESH REDDY  
(2020BTechCSE040)

**FACULTY GUIDE:**  
MR. DIVANSHU JAIN



Institute of Engineering and Technology (IET)  
JK Lakshmi Pat University Jaipur

FEB 2022

<b>Serial No.</b>	<b>Experiment</b>	<b>Page No.</b>
1.	<ul style="list-style-type: none"> <li>• WAP for 8-bit Addition</li> <li>• WAP for 16-bit Addition</li> </ul>	3.
2.	<ul style="list-style-type: none"> <li>• WAP for 8-bit Unsigned Subtraction</li> <li>• WAP for 16-bit Unsigned Subtraction</li> </ul>	7.
3.	Write an ALP for Block transfer (6 byte) using string operation (from location 20000H to 30000H) (also show using MOVSB, MOVSW, CLD and STD mode)	11.
4.	Write an ALP for Block Exchange using string operation (exchange data from location 20000H to 30000H and vice versa)	13.
5.	<ul style="list-style-type: none"> <li>• Write an ALP to multiply two 16-bit numbers. Operands and result in Data Segment.</li> <li>• Write an ALP to divide a 16-bit number by an 8 bit number.</li> </ul>	15.
6.	Write an ALP to add a series of 10 bytes stored in the memory from locations 20,000H to 20,009H. Store the result immediately after the series.	17.
7.	Write an ALP to find the factorial of a number stored at 24,000H in data segment. Store the result at 24,001H and 24,002H.	19.
8.	<ul style="list-style-type: none"> <li>• Write an ALP to invert a block of 10 bytes from Data Segment to Extra Segment.</li> <li>• Write an ALP to get 5 byte of data from the user and store in it memory location 50000H to 50005 H. Find the maximum of it and store it next location.</li> </ul>	21
9.	Write an ALP to SORT a series of 10 numbers from 20,000H in ascending order.	24.
10.	<p>Write an ALP to reverse the string and print the reversed string.  Ex. Input: String: "Geeks for Geeks"  Output: skeeG rof skeeG</p>	25.
11.	Write an ALP to determine how many times “e” exists in “exercise”	27.
12.	Write an ALP to given string is palindrome or not?	28.
13.	Write an ALP to Convert a Decimal Number into Hexadecimal. Assume the Decimal Number is stored at 24000H. Store the result at 24001H.	31.

## EXPERIMENT – 1:

### PART-A:

AIM: WAP for 8-bit Addition

### THEORY:

As we have taken 2 hexadecimal number as follows:

0 F F H

0 A 0 H

-----

1 0 9 F H

It will give a carry of 1, so our answer will be 0 9 F H and carry will be 1.

On emulating the code we'll get the following –

The screenshot shows the emu8086 software interface. On the left, the assembly code is displayed:

```
01 ;Name - Kondrolla Dinesh Reddy
02 ;Roll no. - 2020BTechCSE040
03 ;Batch - B
04
05
06 DATA SEGMENT
07     A DB 0Fh
08     B DB 0Ah
09     SUM DB ?
10     CARRY DB 00h
11 DATA ENDS
12
13 CODE SEGMENT
14     ASSUME CS:CODE, DS:DATA
15 START:
16     MOU AX, DATA
17     MOU DS, AX
18     MOU AL, A
19     MOU BL, B
20     ADD AL, BL
21     JNC SKIP
22     INC CARRY
23     SKIP: MOU SUM, AL
24     MOU AH, 4Ch
25     INT 21h
26     ;INT 03h
27 CODE ENDS
28 END START
29
30
31
32
```

In the center, there are two windows: one showing the original source code and another showing the emulator window. The emulator window displays the registers and memory dump. The registers window shows:

Registers	H	L
AX	4C	9F
BX	0B	A0
CX	00	2B
DX	00	00
CS	F400	
IP	0204	
SS	0710	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

The memory dump window shows the memory starting at address F400:0204. The first few bytes are:

Address	Value	Content
F400:0204	CF 2B7	INT 021h
F4205: 00 000	NULL	ADD BX + SI1, AL
F4206: 00 000	NULL	ADD BX + SI1, AL
F4207: 00 000	NULL	ADD BX + SI1, AL
F4208: 00 000	NULL	ADD BX + SI1, AL
F4209: 00 000	NULL	ADD BX + SI1, AL
F420A: 00 000	NULL	ADD BX + SI1, AL
F420B: 00 000	NULL	ADD BX + SI1, AL
F420C: 00 000	NULL	ADD BX + SI1, AL
F420D: 00 000	NULL	ADD BX + SI1, AL
F420E: 00 000	NULL	ADD BX + SI1, AL
F420F: 00 000	NULL	ADD BX + SI1, AL
F4210: 00 000	NULL	ADD BX + SI1, AL
F4211: 00 000	NULL	ADD BX + SI1, AL
F4212: 00 000	NULL	ADD BX + SI1, AL
F4213: 00 000	NULL	ADD BX + SI1, AL
F4214: 00 000	NULL	ADD BX + SI1, AL
F4215: 00 000	NULL	...

The status bar at the bottom shows: line: 3 col: 28 drag a file here to open ENG IN 26-02-2022

```
01 ;Name - Kondrolla Dinesh Reddy
02 ;Roll no. - 2020BTechCSE040
03 ;Batch - B
04
05 DATA SEGMENT
06     A DB 0FFH
07     B DB 0A0H
08     SUM DB ?
09     CARRY DB 00H
10     DATA ENDS
11
12 CODE SEGMENT
13     ASSUME CS:CODE , DS:DATA
14     START:
15         MOU AX,DATA
16         MOU DS,AX
17         MOU AL,A
18         MOU BL,B
19         ADD AL,BL
20         JNC SKIP
21         INC CARRY
22         SKIP: MOU SUM,AL
23         MOU AH, 4CH
24         INT 21H
25         ;INT 03h
26         CODE ENDS
27 END START
28
29
30
31
32
```

## PART-B:

**AIM:** WAP for 16-bit Addition

## THEORY:

As we have taken 2 hexadecimal number as follows:

0 F F A 0 H

0 A A 1 0 H

-----

1 0 A 9 B 0 H

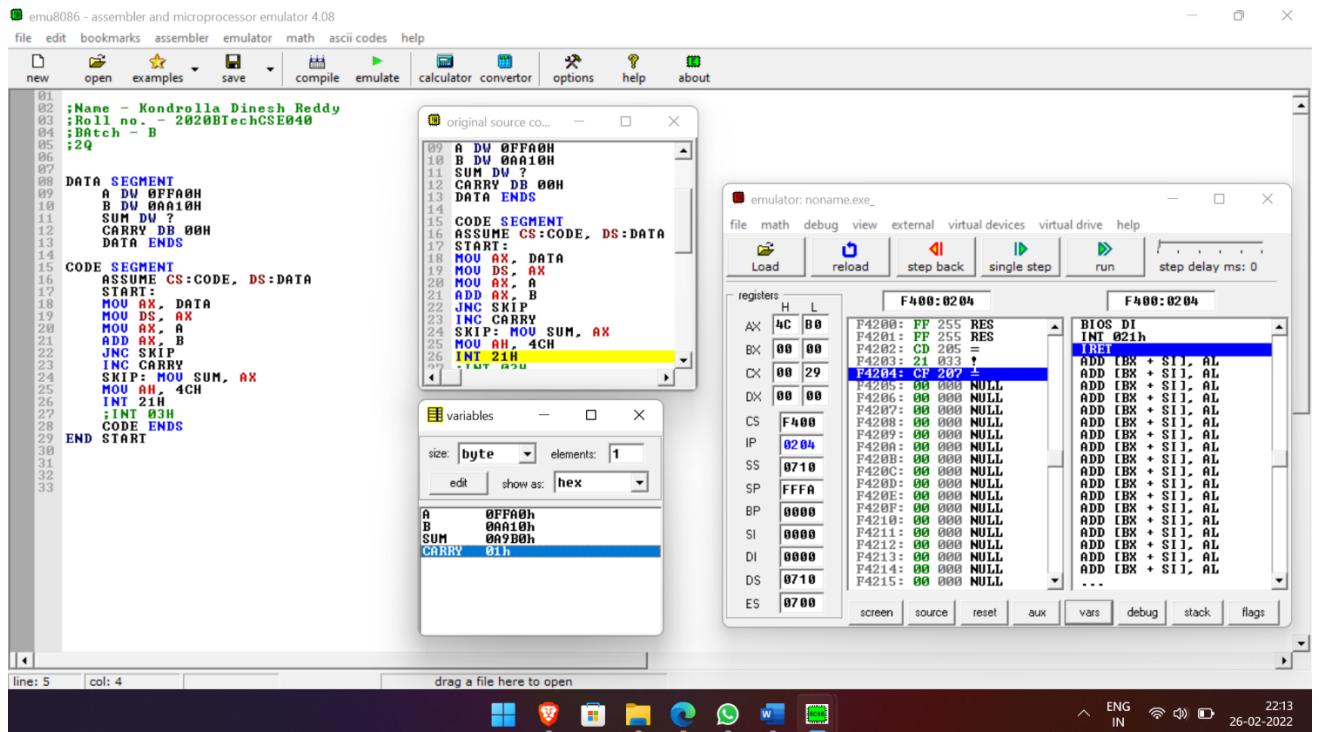
It will give a carry of 1, so our answer will be 0 A 9 B 0 H and carry will be 1.

On emulating the code we'll get the following –

```
;Name - Kondrolla Dinesh Reddy
;Roll no. - 2020BTechCSE040
;Batch - B
;2Q

DATA SEGMENT
    A DW 0FFA0H
    B DW 0AA10H
    SUM DW ?
    CARRY DB 00H
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOU AX, DATA
    MOU DS, AX
    MOU AX, A
    ADD AX, B
    JNC SKIP
    INC CARRY
    SKIP: MOU SUM, AX
    MOU AH, 4CH
    INT 21H
    ;INT 03H
CODE ENDS
END START
```



## **EXPERIMENT – 2:**

### **PART-A:**

**AIM:** WAP for 8-bit Unsigned Subtraction

### **THEORY:**

As we have taken 2 hexadecimal number as follows:

0 3 6 H

0 4 2 H

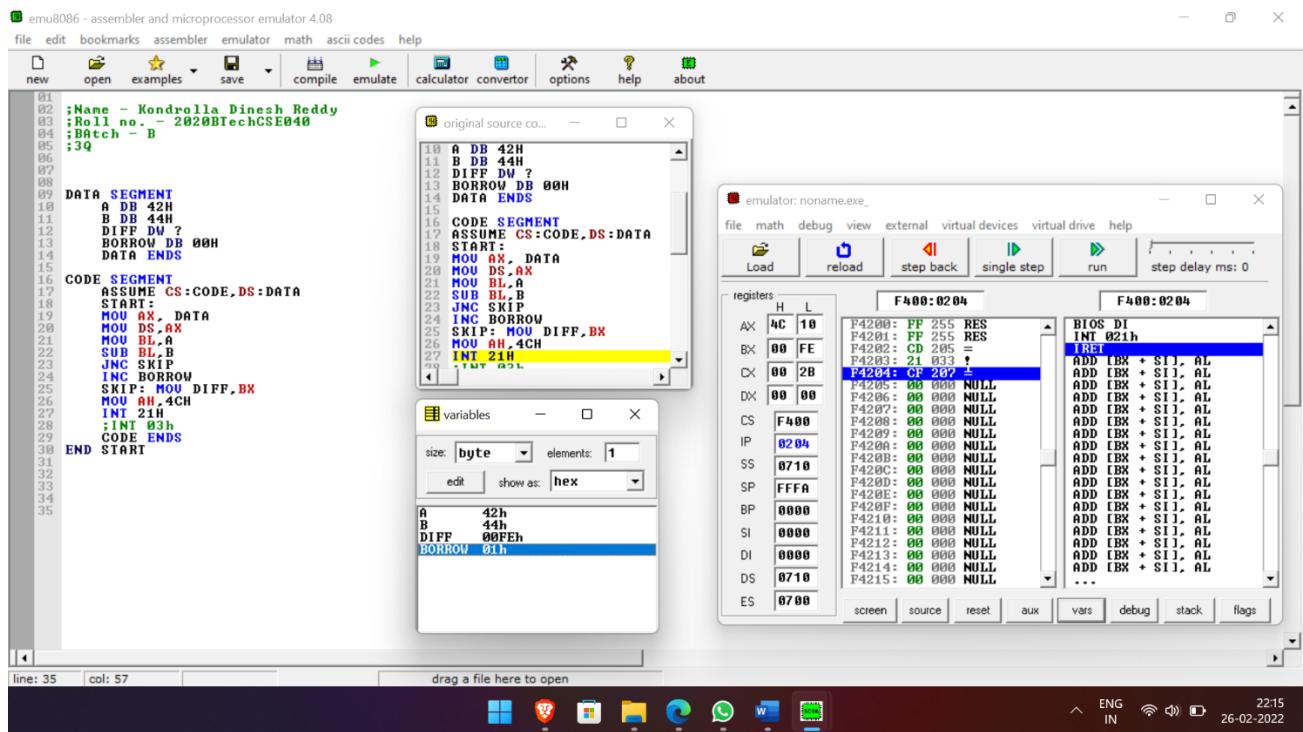
-----

1 0 F 4 H

It will take a borrow of 1, so our answer will be 0 F 4 H and borrow will be 1.

On emulating the code we'll get the following –

```
01 ;Name - Kondrolla Dinesh Reddy
02 ;Roll no. - 2020BTechCSE040
03 ;Batch - B
04 ;3Q
05
06
07
08
09 DATA SEGMENT
10 A DB 42H
11 B DB 44H
12 DIFF DW ?
13 BORROW DB 00H
14 DATA ENDS
15
16 CODE SEGMENT
17 ASSUME CS:CODE, DS:DATA
18 START:
19 MOV AX, DATA
20 MOV DS, AX
21 MOV BL, A
22 SUB BL, B
23 JNC SKIP
24 INC BORROW
25 SKIP: MOU DIFF, BX
26 MOU AH, 4CH
27 INT 21H
28 ;INT 03h
29 CODE ENDS
30 END START
31
```



## PART-B

**AIM:** WAP for 16-bit Unsigned Subtraction

### THEORY:

As we have taken 2 hexadecimal number as follows:

0 F F A 0 H

0 A A 1 0 H

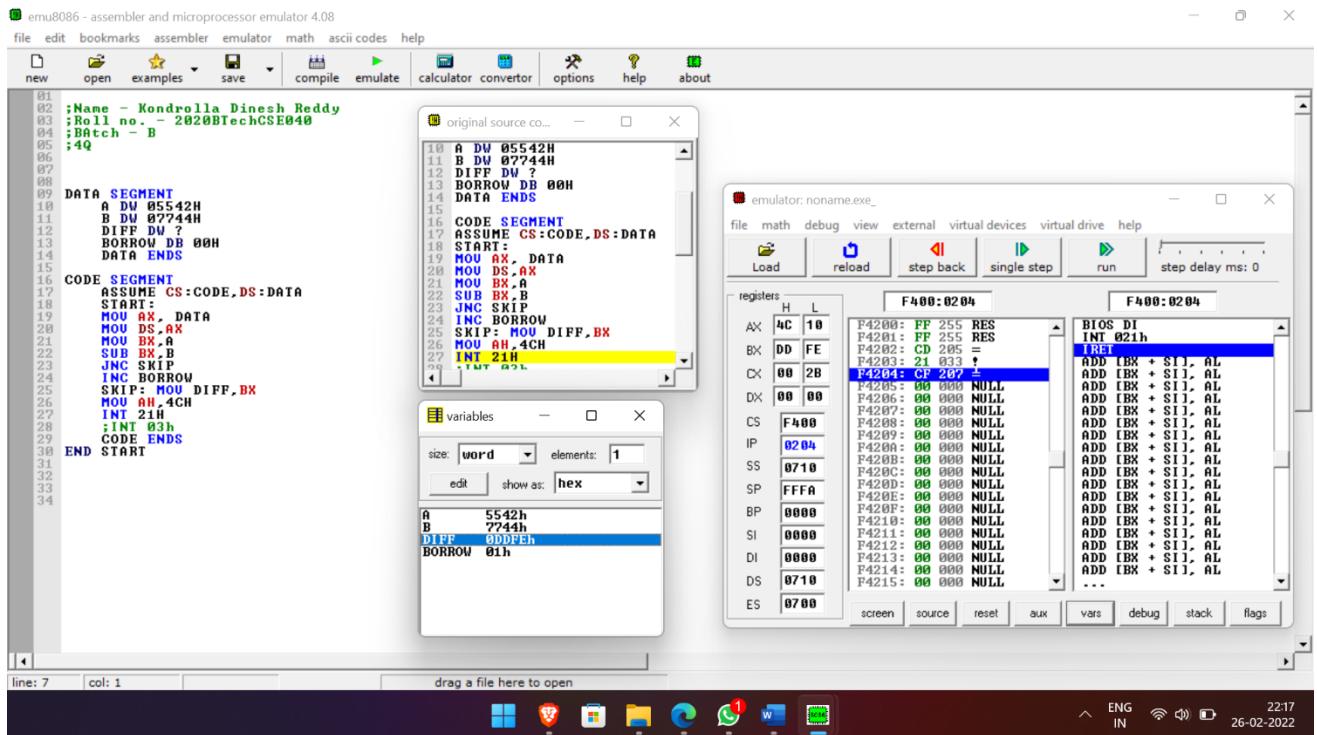
-----

0 5 5 9 0 H

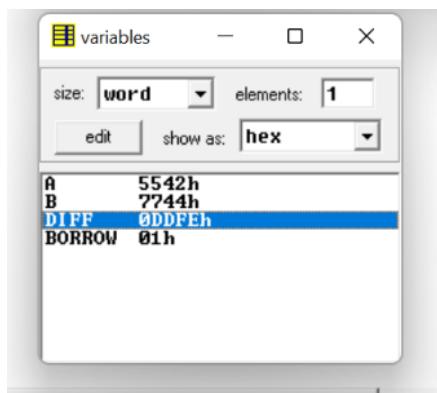
It will take a borrow of 1, so our answer will be 0 5 5 9 0 H and borrow will be 0.

On emulating the code we'll get the following –

```
01 ;Name - Kondrolla Dinesh Reddy
02 ;Roll no. - 2020BTechCSE040
03 ;Batch - B
04 ;4Q
05
06
07
08
09 DATA SEGMENT
10 A DW 05542H
11 B DW 07744H
12 DIFF DW ?
13 BORROW DB 00H
14 DATA ENDS
15
16 CODE SEGMENT
17 ASSUME CS:CODE, DS:DATA
18 START:
19 MOU AX, DATA
20 MOU DS, AX
21 MOU BX, A
22 SUB BX, B
23 JNC SKIP
24 INC BORROW
25 SKIP: MOU DIFF, BX
26 MOU AH, 4CH
27 INT 21H
28 ;INT 03h
29 CODE ENDS
30 END START
31
```



## RESULT:



## EXPERIMENT – 3:

**AIM:** WAP for 16-bit Unsigned Subtraction

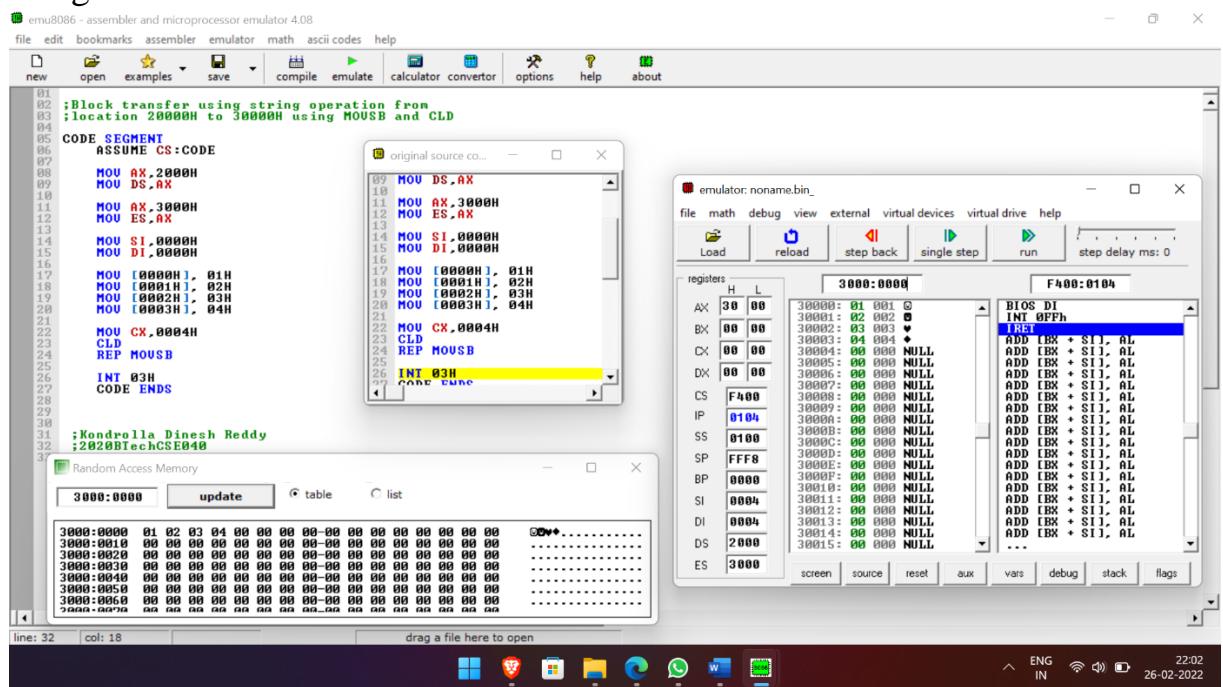
### THEORY:

On emulating and running the code, we get the following windows and after we'll click the button named memory in aux to check whether the block of data has been transferred to the destination or not by checking the data of it using address.

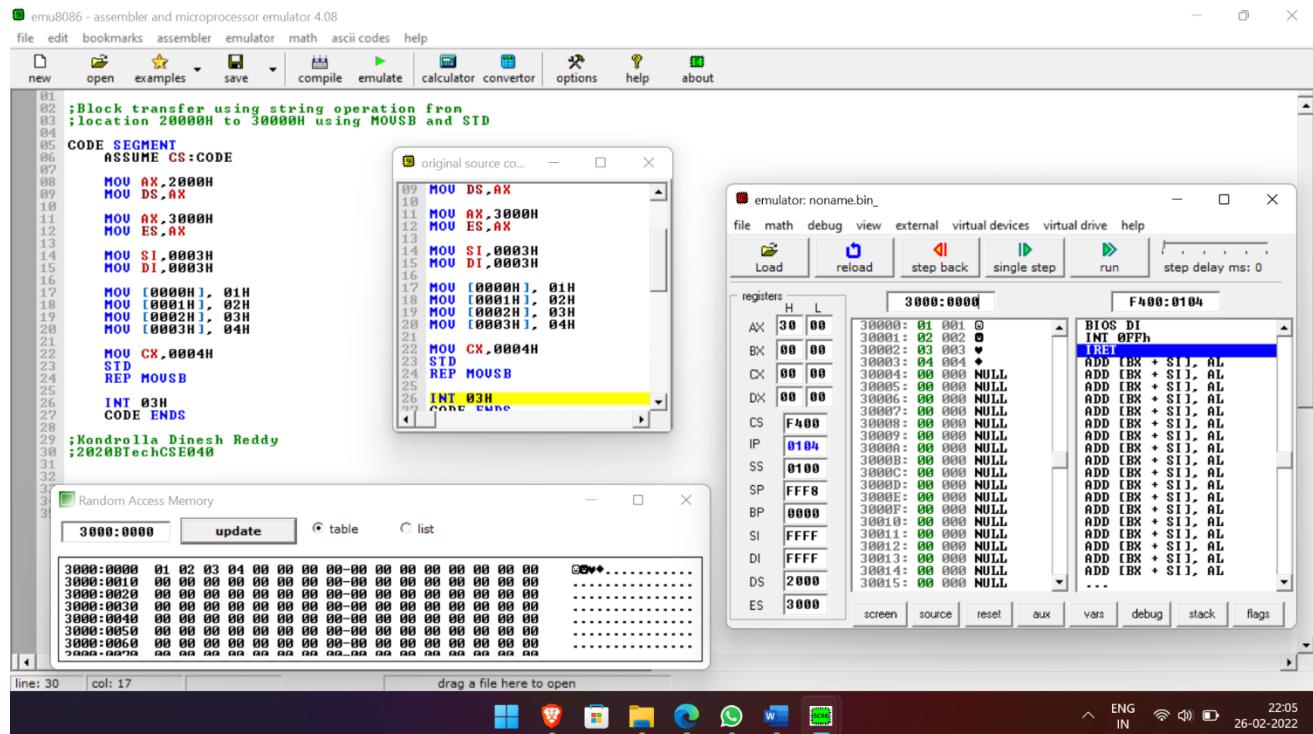
For the whole data transfer the data should be there in the source (2000) and destination (3000).

### CODE:

- Block transfer using string operation from location 20000H to 30000H using MOVS



- Block transfer using string operation from location 20000H to 30000H using MOVSB and STD.



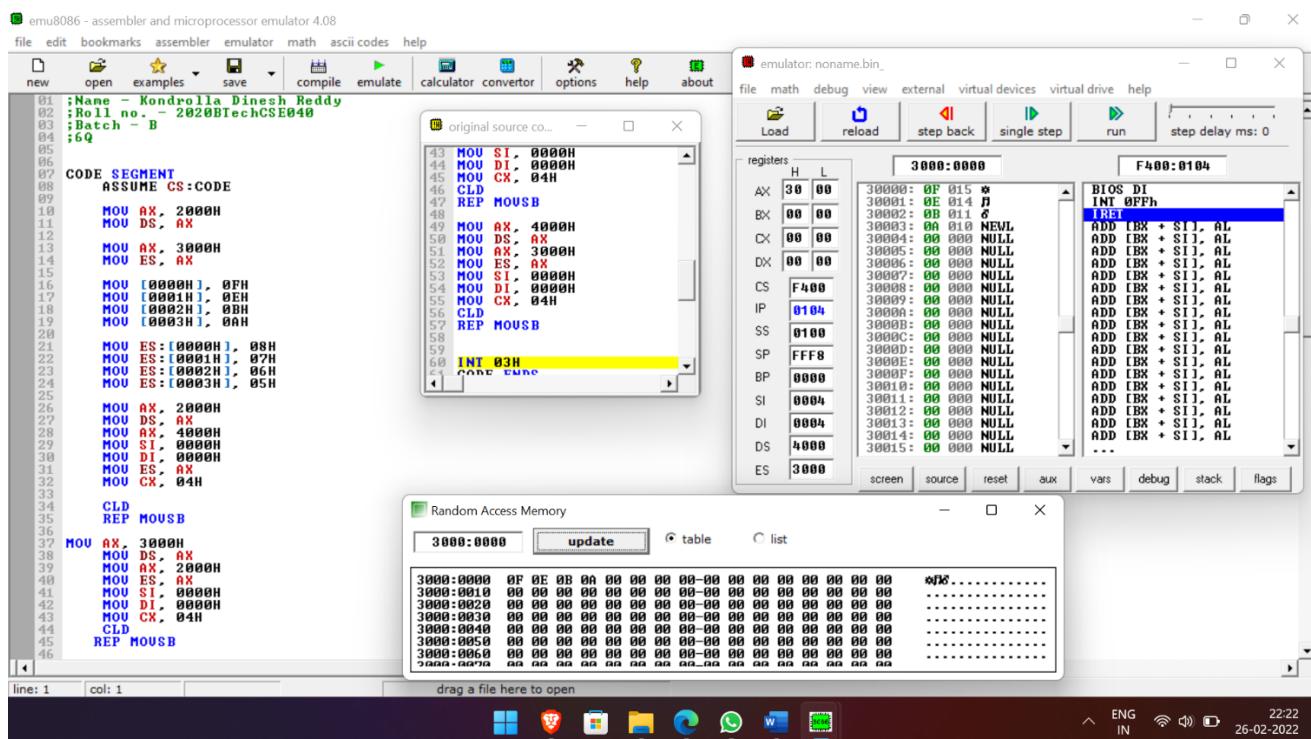
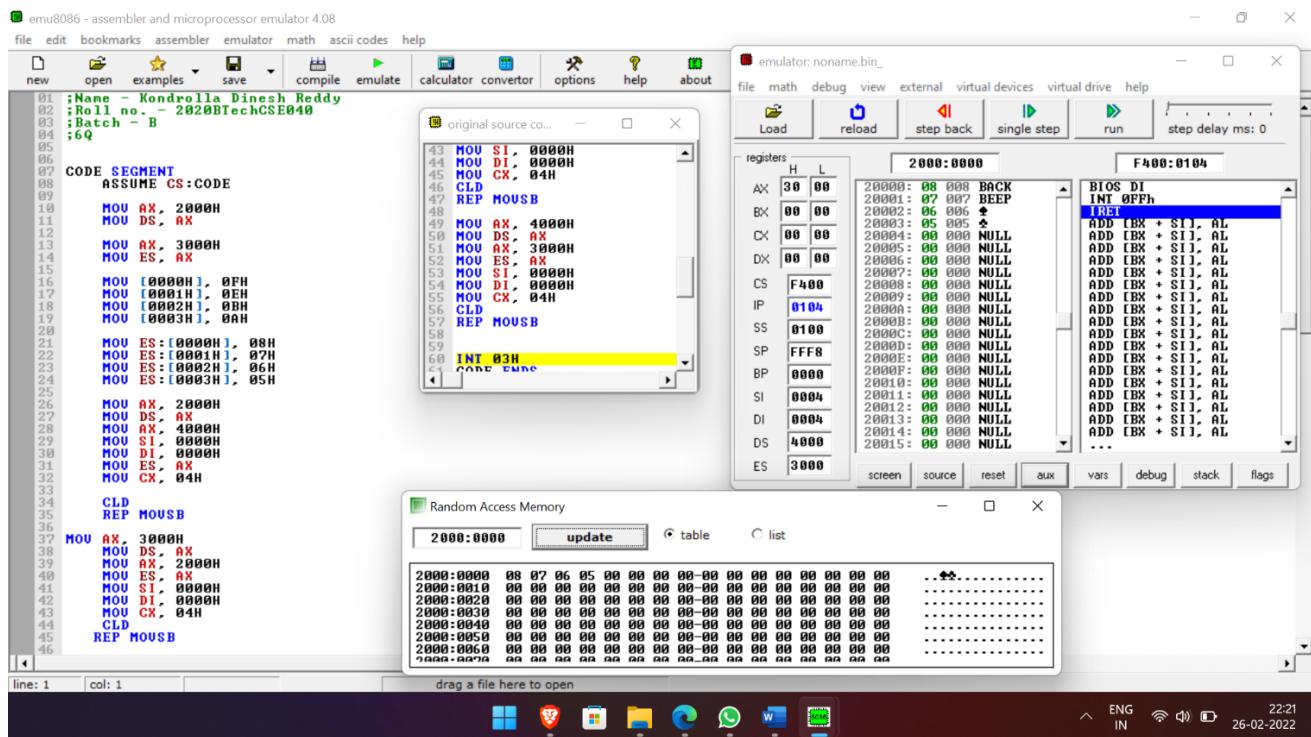
## EXPERIMENT – 4:

**AIM:** Block Exchange using String Operation

### THEORY:

Block exchange program has exchanged the data of 20000H and 30000H which means that the data of 20000H has been stored in the 30000H and the data of the 30000H has been stored to 20000H.

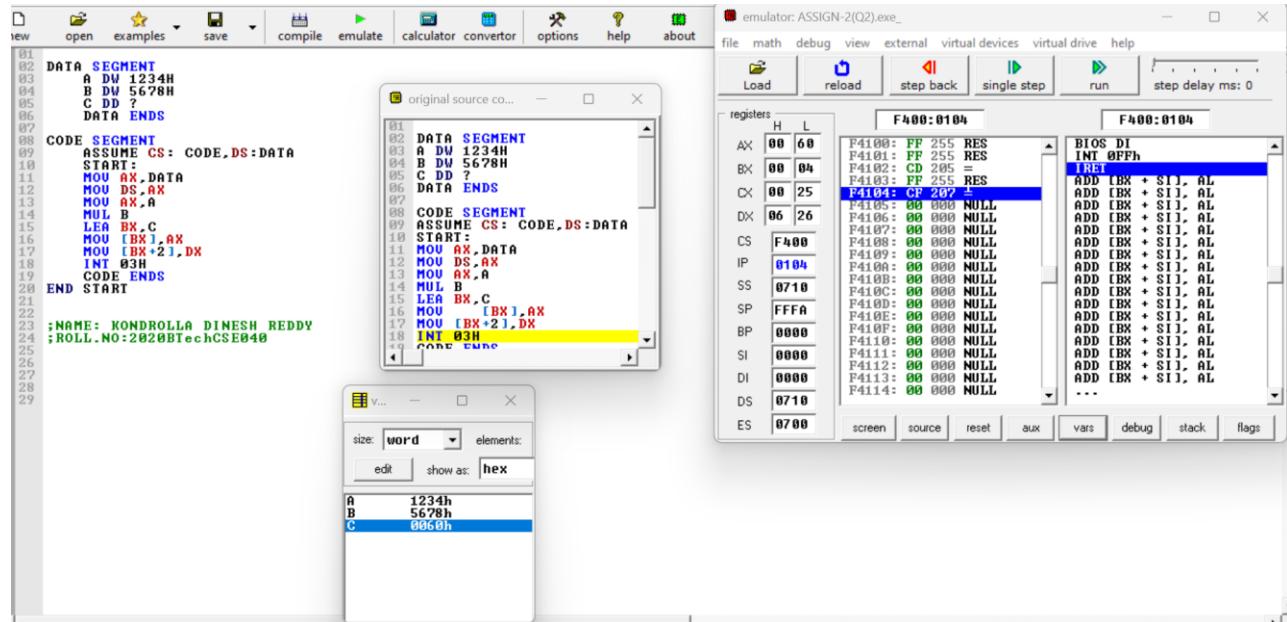
```
01 ;Name - Kondrolla Dinesh Reddy
02 ;Roll no. - 2020BTechCSE040
03 ;Batch - B
04 ;6Q
05
06
07 CODE SEGMENT
08 ASSUME CS:CODE
09
10 MOU AX, 2000H
11 MOU DS, AX
12
13 MOU AX, 3000H
14 MOU ES, AX
15
16 MOU [0000H], 0FH
17 MOU [0001H], 0EH
18 MOU [0002H], 0BH
19 MOU [0003H], 0AH
20
21 MOU ES:[0000H], 08H
22 MOU ES:[0001H], 07H
23 MOU ES:[0002H], 06H
24 MOU ES:[0003H], 05H
25
26 MOU AX, 2000H
27 MOU DS, AX
28 MOU AX, 4000H
29 MOU SI, 0000H
30 MOU DI, 0000H
31 MOU ES, AX
32 MOU CX, 04H
33
34 CLD
35 REP MOUSB
36
37 MOU AX, 3000H
38 MOU DS, AX
39 MOU AX, 2000H
40 MOU ES, AX
41 MOU SI, 0000H
42 MOU DI, 0000H
43 MOU CX, 04H
44 CLD
45 REP MOUSB
46
```



## EXPERIMENT – 5:

**AIM:** Write an ALP to multiply two 16-bit numbers. Operands and result in Data Segment.

**CODE:**



**RESULT:**

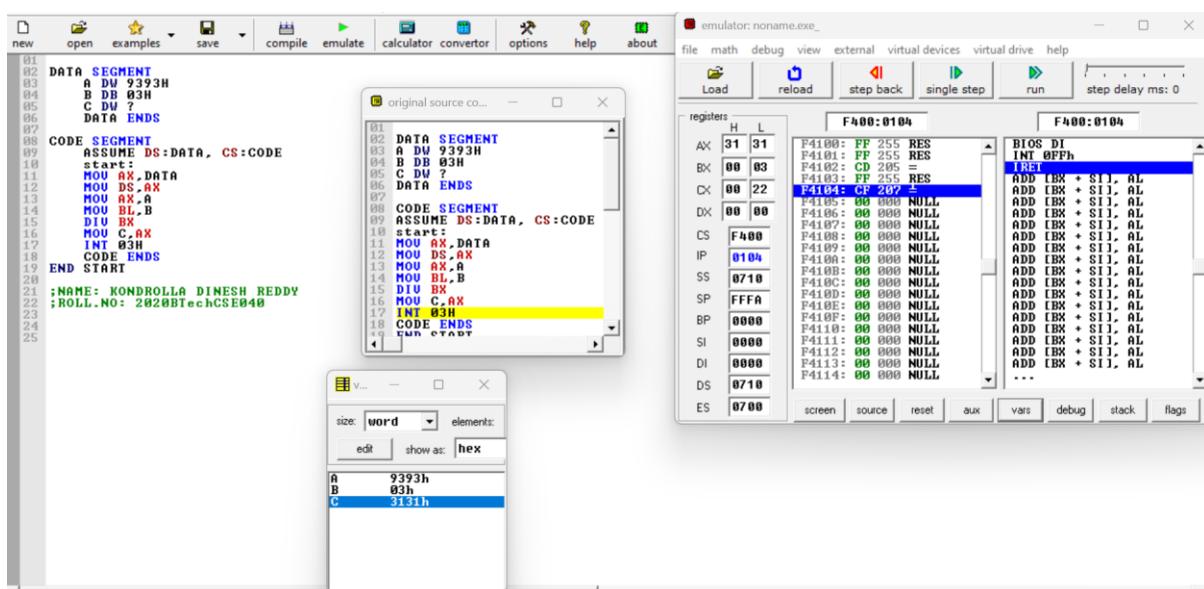


## PART-B:

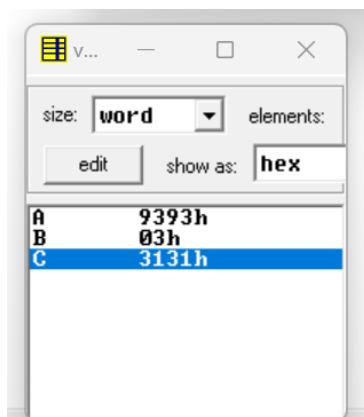
**AIM:** Write an ALP to divide a 16-bit number by an 8 bit number.

### CODE:

```
01 DATA SEGMENT
02     A DW 9393H
03     B DB 03H
04     C DW ?
05     DATA ENDS
06
07 CODE SEGMENT
08     ASSUME DS:DATA, CS:CODE
09     start:
10         MOU AX,DATA
11         MOU DS,AX
12         MOU AX,A
13         MOU BL,B
14         DIU BX
15         MOU C,AX
16         INT 03H
17     CODE ENDS
18 END START
19
20 ;NAME: KONDROLLA DINESH REDDY
21 ;ROLL.NO: 2020BTechCSE040
22
23
24
```



### RESULT:



## EXPERIMENT – 6:

**AIM:** Write an ALP to add a series of 10 bytes stored in the memory from locations 20,000H to 20,009H. Store the result immediately after the series.

### CODE:

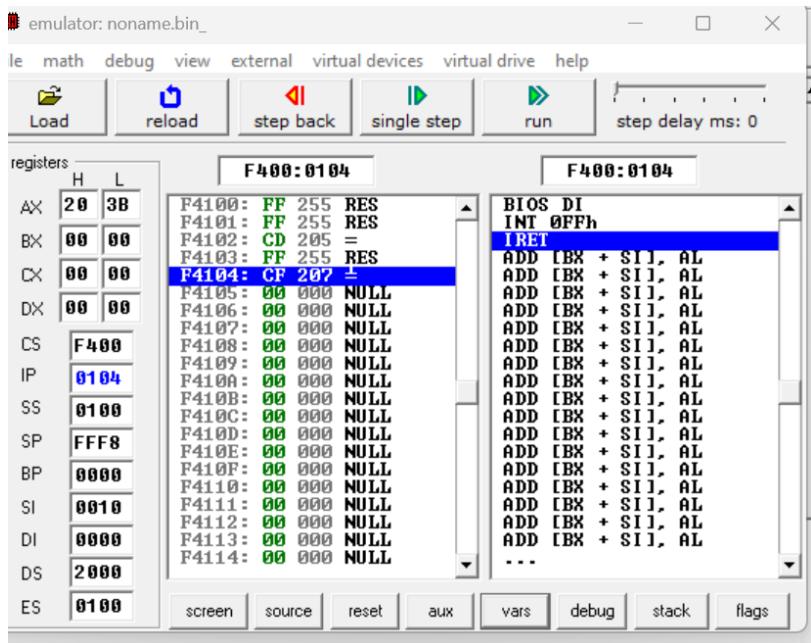
```
01  CODE SEGMENT
02  ASSUME CS: CODE
03  MOV AX, 2000H
04  MOV DS, AX
05  MOV SI, 0000H
06  MOV [0000H], 04H
07  MOV [0001H], 06H
08  MOV [0002H], 08H
09  MOV [0003H], 03H
10  MOV [0004H], 02H
11  MOV [0005H], 09H
12  MOV [0006H], 07H
13  MOV [0007H], 03H
14  MOV [0008H], 09H
15  MOV [0009H], 08H
16
17
18  MOV CX, 10H
19  MOV AL, 00H
20
21  BACK: ADD AL, [SI]
22  JNC SKIP
23  MOV AL, [SI]
24
25  SKIP: INC SI
26  LOOP BACK
27  MOV [SI], AL
28  INT 03H
29
30  CODE ENDS
31 ;NAME: KONDROLLA DINESH REDDY
32 ;ROLL.NO: 2020BTechCSE040
33
34
35
36
```

The screenshot shows a debugger interface with the following details:

- Registers:** AX, BX, CX, DX, CS, IP, SS, SP, BP, SI, DI, DS, ES.
- Memory Dump:** F400:0104 to F400:0104, showing the assembly code:

```
F400:0104: FF 255 RES
F400:0105: FF 255 RES
F400:0106: CD 205 =
F400:0107: FF 255 RES
F400:0108: FF 000 NULL
F400:0109: FF 000 NULL
F400:010A: FF 000 NULL
F400:010B: FF 000 NULL
F400:010C: FF 000 NULL
F400:010D: FF 000 NULL
F400:010E: FF 000 NULL
F400:010F: FF 000 NULL
F400:0110: FF 000 NULL
F400:0111: FF 000 NULL
F400:0112: FF 000 NULL
F400:0113: FF 000 NULL
F400:0114: FF 000 NULL
```
- Stack:** BIOS DI INT 0FFh, IRET.
- Code View:** Shows the assembly code with the instruction at address F400:0104 highlighted.

## **RESULT:**

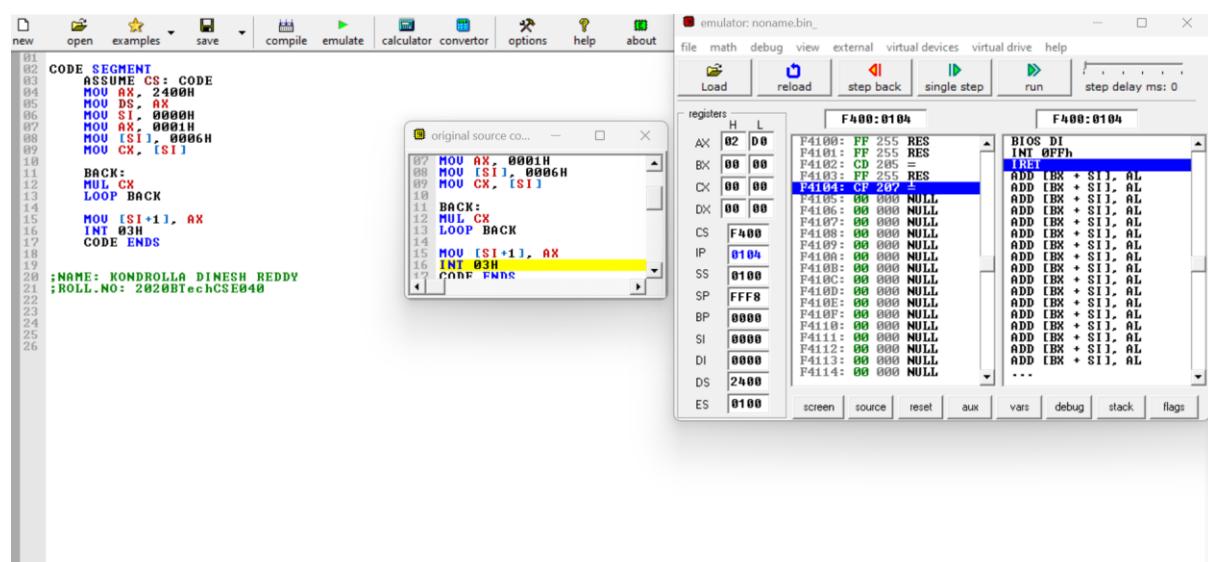


## EXPERIMENT – 7:

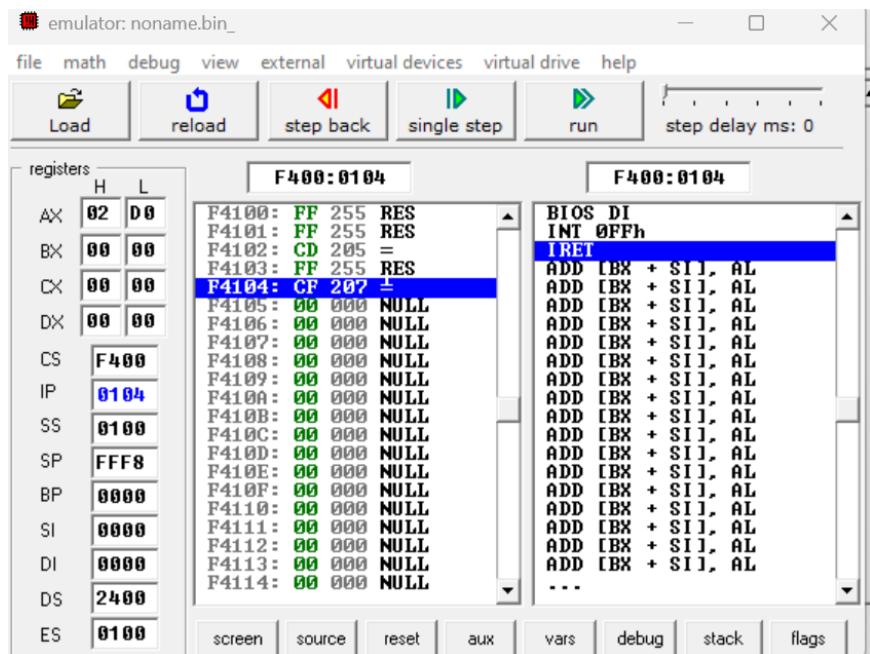
**AIM:** Write an ALP to find the factorial of a number stored at 24,000H in data segment. Store the result at 24,001H and 24,002H.

### CODE:

```
01  CODE SEGMENT
02  ASSUME CS: CODE
03  MOU AX, 2400H
04  MOU DS, AX
05  MOU SI, 0000H
06  MOU AX, 0001H
07  MOU [SI], 0006H
08  MOU CX, [SI]
09
10
11 BACK:
12 MUL CX
13 LOOP BACK
14
15 MOU [SI+1], AX
16 INT 03H
17 CODE ENDS
18
19
20 ;NAME: KONDROLLA DINESH REDDY
21 ;ROLL.NO: 2020BTechCSE040
22
23
```



## RESULT:



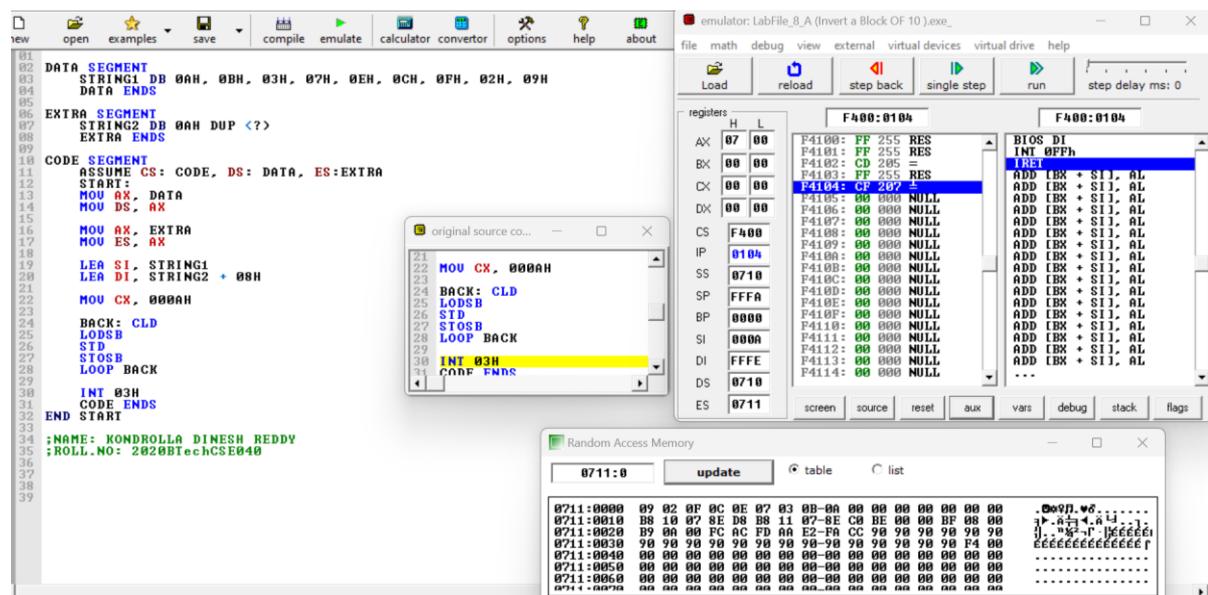
## EXPERIMENT-8:

### PART-A

**AIM:** Write an ALP to invert a block of 10 bytes from Data Segment to Extra Segment.

### CODE:

```
01 DATA SEGMENT
02     STRING1 DB 0AH, 0BH, 03H, 07H, 0EH, 0CH, 0FH, 02H, 09H
03     DATA ENDS
04
05 EXTRA SEGMENT
06     STRING2 DB 0AH DUP <?>
07     EXTRA ENDS
08
09 CODE SEGMENT
10     ASSUME CS: CODE, DS: DATA, ES:EXTRA
11     START:
12         MOU AX, DATA
13         MOU DS, AX
14
15         MOU AX, EXTRA
16         MOU ES, AX
17
18         LEA SI, STRING1
19         LEA DI, STRING2 + 08H
20
21         MOU CX, 000AH
22
23 BACK: CLD
24 LODSB
25 STD
26 STOSB
27 LOOP BACK
28
29 INT 03H
30
31 CODE ENDS
32 END START
33
34 ;NAME: KONDROLLA DINESH REDDY
35 ;ROLL.NO: 2020BTechCSE040
36
37
```



## RESULT:

Random Access Memory			
0711:0	update	table	list
0711:0000	09 02 0F 0C 0E 07 03 0B-0A 00 00 00 00 00 00 00 00	. . . . .	
0711:0010	B8 10 07 8E D8 B8 11 07-8E C0 BE 00 00 00 BF 08 00	► . . . . .	
0711:0020	B9 0A 00 FC AC FD AA E2-FA CC 90 90 90 90 90 90 90	. . . . .	
0711:0030	90 90 90 90 90 90 90 90-90 90 90 90 90 90 90 F4 00	E E E E E E E E E E E E E E E E E E	
0711:0040	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	. . . . .	
0711:0050	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	. . . . .	
0711:0060	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	. . . . .	
0711:0070	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	. . . . .	

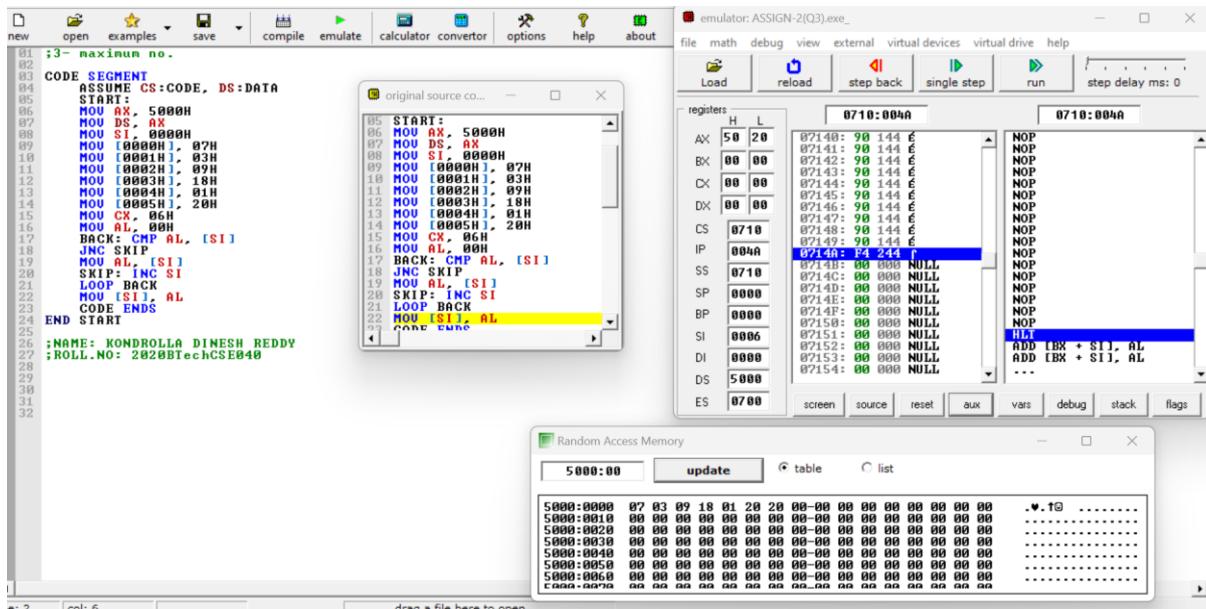
## PART-B:

**AIM:** Write an ALP to get 5 byte of data from the user and store in it memory location 50000H to 50005 H. Find the maximum of it and store it next location.

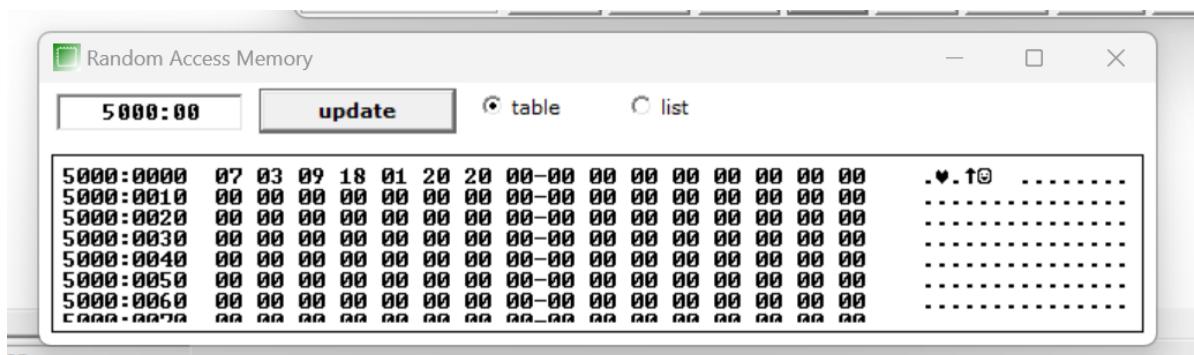
## CODE:

```
new open examples save compile
;3- maximum no.

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
MOU AX, 5000H
MOU DS, AX
MOU SI, 0000H
MOU [0000H], 07H
MOU [0001H], 03H
MOU [0002H], 09H
MOU [0003H], 18H
MOU [0004H], 01H
MOU [0005H], 20H
MOU CX, 06H
MOU AL, 00H
BACK: CMP AL, [SI]
JNC SKIP
MOU AL, [SI]
SKIP: INC SI
LOOP BACK
MOU [SI], AL
CODE ENDS
END START
;NAME: KONDROLLA DINESH REDDY
;ROLL.NO: 2020BTechCSE040
25
26
27
28
29
30
31
```



## RESULT:



## EXPERIMENT – 9:

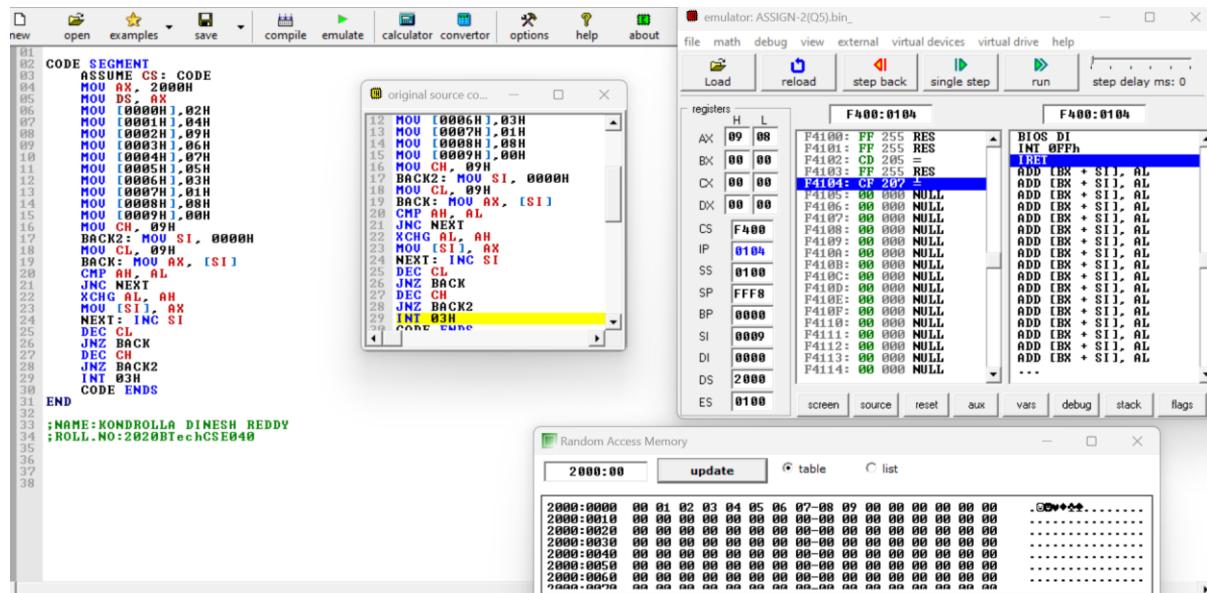
**AIM:** Write an ALP to SORT a series of 10 numbers from 20,000H in ascending order.

### CODE:

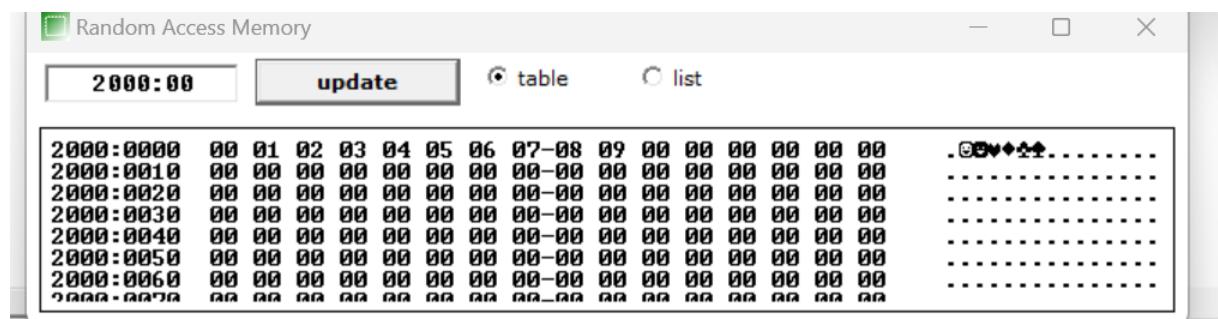
```

01  CODE SEGMENT
02      ASSUME CS: CODE
03      MOU AX, 2000H
04      MOU DS, AX
05      MOU [0000H], 02H
06      MOU [0001H], 04H
07      MOU [0002H], 06H
08      MOU [0003H], 08H
09      MOU [0004H], 0AH
10      MOU [0005H], 05H
11      MOU [0006H], 03H
12      MOU [0007H], 01H
13      MOU [0008H], 08H
14      MOU [0009H], 00H
15      MOU [000AH], 09H
16      MOU CH, 09H
17      BACK2: MOU SI, 0000H
18      MOU CL, 09H
19      BACK: MOU AX, [SI]
20      CMP AH, AL
21      JNC NEXT
22      XCHG AL, AH
23      MOU [SI], AX
24      NEXT: INC SI
25      DEC CL
26      INT BACK
27      DEC CH
28      JNZ BACK2
29      INT 03H
30      CODE ENDS
31
32
33 ;NAME:KONDROLLA DINESH REDDY
34 ;ROLL.NO:2020BTechCSE040
35
36

```



### RESULT:



## EXPERIMENT – 10:

**AIM:** Write an ALP to reverse the string and print the reversed string.

Ex. Input: String: "Geeks for Geeks"

Output: skeeG rof skeeG

### CODE:

The screenshot shows the Microsoft Macro Assembler (MASM) interface. The assembly code is as follows:

```
01 DATA SEGMENT
02     STR1 DB "geeks_for_geeks$"
03     STR2 DB 28 DUP ('$')
04     M1 DB 10,13,'STORED STRING IN MEMORY IS : $'
05     M2 DB 10,13,'REVERSE STRING IS : $'
06 DATA ENDS
07 DISPLAY MACRO MSG
08     MOU AH,9
09     LEA DX,MSG
10     INT 21H
11 ENDM
12 CODE SEGMENT
13     ASSUME CS:CODE, DS:DATA
14 START:
15     MOU AX,DATA
16     MOU DS,AX
17     DISPLAY M1
18     DISPLAY STR1
19     LEA SI,STR2
20     LEA DI,STR1
21     ADD DI,14
22     MOU CX,15
23 REVERSE:
24     MOU AL,[DI]
25     MOU [SI],AL
26     INC SI
27     DEC DI
28     LOOP REVERSE
29     DISPLAY M2
30     DISPLAY STR2
31     MOU AH,4CH
32     INT 21H
33 CODE ENDS
34 END START
35
36 ;NAME: KONDROLLA DINESH REDDY
37 ;ROLL.NO:2020BTechCES040
38
39
40
41
42
43
```

The registers window shows:

STR1	67h
STR2	73h
M1	0Ah
M2	0Ah

The memory dump window shows:

size: byte	elements:
edit	show as: hex
STR1	67h
STR2	73h
M1	0Ah
M2	0Ah

The emulator screen displays:

```
emulator screen (80x4 chars)
STORED STRING IN MEMORY IS: geeks_for_geeks
REVERSE STRING IS: skeeg_rof_skeeg
```

At the bottom, there are buttons for 'clear screen' and 'change font', and a status bar showing 'e: 38' and '0/16'.

The screenshot shows a debugger interface with the following components:

- Assembly View:** Displays the assembly code for the program. It includes labels like .DATA, .CODE, and .START, and instructions such as MOU AX,DATA, LEA DS,AX, and various DISPLAY, INC, DEC, ADD, and MOU instructions.
- Registers View:** Shows the CPU registers (AX, BX, CX, DX, CS, IP, SS, BP, SI, DI, DS, ES) with their current values and memory addresses.
- Stack View:** Shows the stack contents, which include the strings "geeks\_for\_geeks" and "skeeg\_rof\_skeeg".
- Memory Dump View:** Displays the memory dump at address 0710:0000, showing the byte sequence for the stored and reverse strings.

## RESULT:

The screenshot shows the emulator screen displaying the output of the assembly program. The screen shows:

- Line numbers 39-43.
- A yellow box labeled "SCR" followed by "emulator screen (80x4 chars)".
- The text "STORED STRING IN MEMORY IS: geeks\_for\_geeks" and "REVERSE STRING IS: skeeg\_rof\_skeeg".
- Buttons for "clear screen" and "change font".
- A status bar at the bottom showing "e: 38" and "0/16".

## EXPERIMENT – 11:

**AIM:** Write an ALP to determine how many times “e” exists in “exercise”?

### CODE:

```
DATA SEGMENT
    STRING DB 'EXERCISE'
    COUNT DB 00H
    DATA ENDS

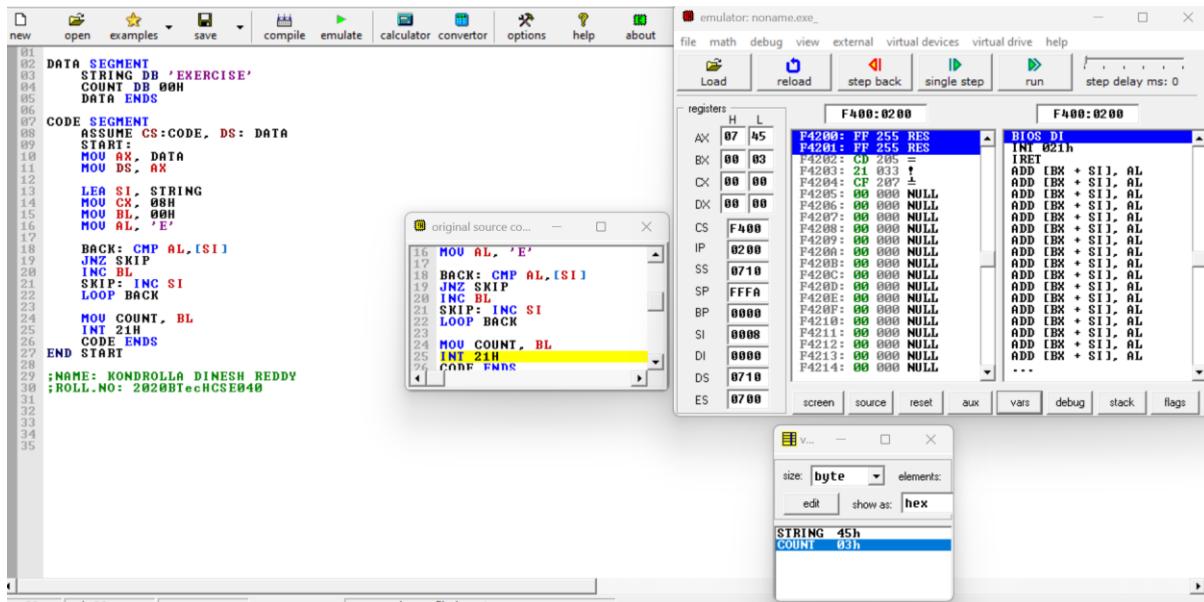
CODE SEGMENT
    ASSUME CS:CODE, DS: DATA
    START:
        MOU AX, DATA
        MOU DS, AX

        LEA SI, STRING
        MOU CX, 08H
        MOU BL, 00H
        MOU AL, 'E'

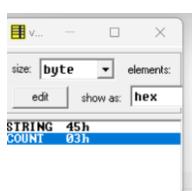
        BACK: CMP AL,[SI]
        JNZ SKIP
        INC BL
        SKIP: INC SI
        LOOP BACK

        MOU COUNT, BL
        INT 21H
    CODE ENDS
END START

;NAME: KONDROLLA DINESH REDDY
;ROLL.NO: 2020BTechCSE040
31
```



### RESULT:



## **EXPERIMENT – 12:**

**AIM:** Write an ALP to given string is palindrome or not?

### **CODE:**

```
DATA SEGMENT
    STRING1 DB "PNKDRNITKTINRDKNP"
    LEN EQU $-STRING1 ;FOR FINDING LENGTH
    PAL DB 00H
    DATA ENDS

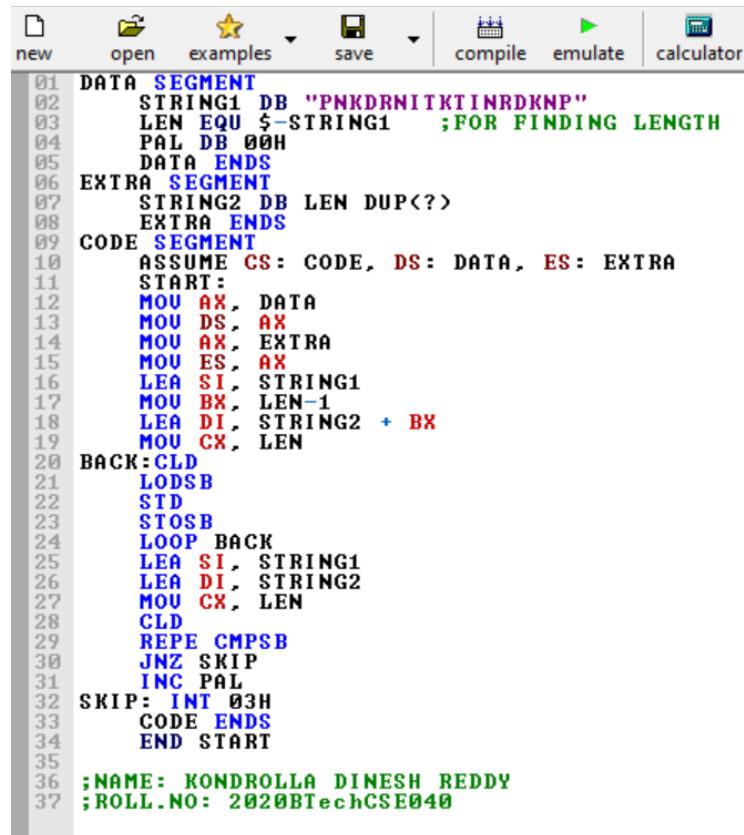
EXTRA SEGMENT
    STRING2 DB LEN DUP(?)
    EXTRA ENDS

CODE SEGMENT
    ASSUME CS: CODE, DS: DATA, ES: EXTRA
    START:
        MOV AX, DATA
        MOV DS, AX
        MOV AX, EXTRA
        MOV ES, AX
        LEA SI, STRING1
        MOV BX, LEN-1
        LEA DI, STRING2 + BX
        MOV CX, LEN

    BACK:CLD
        LODSB
        STD
        STOSB
        LOOP BACK
        LEA SI, STRING1
        LEA DI, STRING2
        MOV CX, LEN
        CLD
        REPE CMPSB
        JNZ SKIP
```

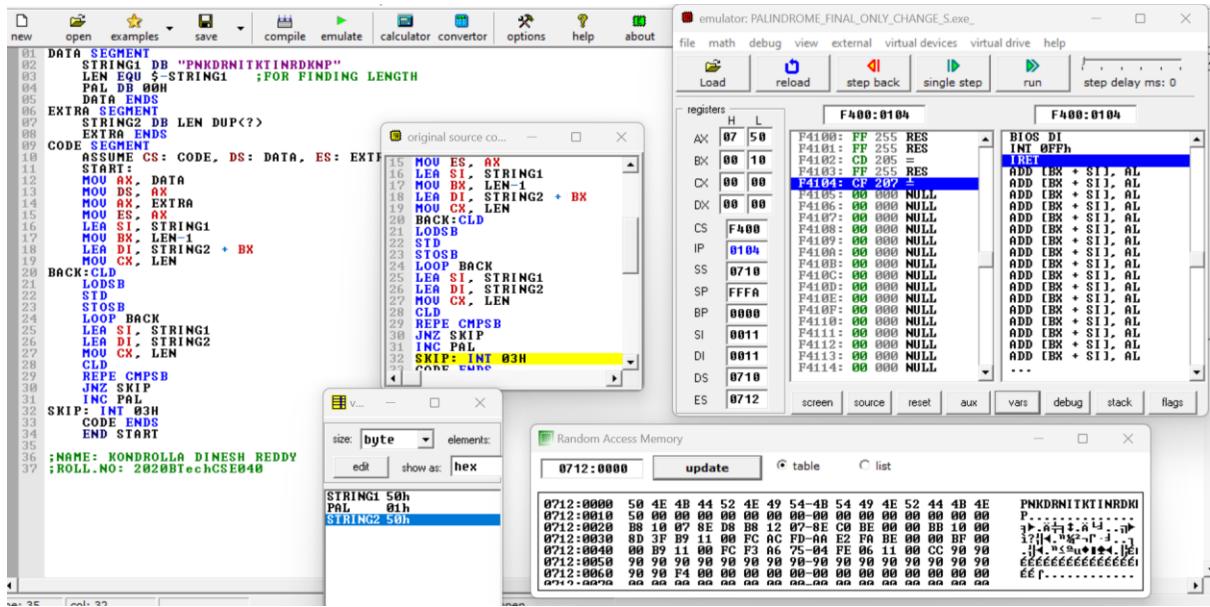
```
INC PAL  
SKIP: INT 03H  
CODE ENDS  
END START
```

;NAME: KONDROLLA DINESH REDDY  
;ROLL.NO: 2020BTechCSE040

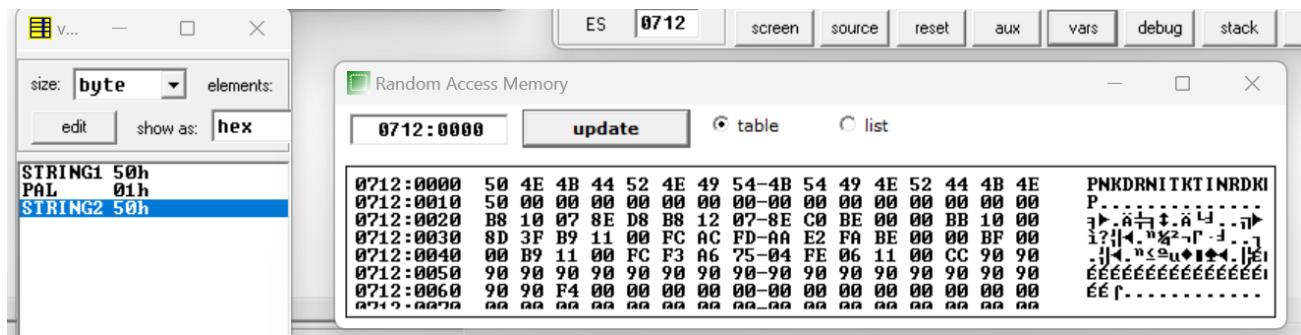


The image shows a screenshot of a Microsoft Notepad window with a dark theme. The window title bar includes icons for new, open, examples, save, compile, emulate, and calculator. The main content area contains assembly language code. The code defines segments for DATA, EXTRA, and CODE, and includes instructions for moving strings between memory segments and performing string operations like copying and comparing. It also includes a loop for string reversal and handles the INT 03H interrupt. At the bottom of the code, there are two comments identifying the author and roll number.

```
01 DATA SEGMENT  
02 STRING1 DB "PNKDRNITKTINRDKNP"  
03 LEN EQU $-STRING1 ;FOR FINDING LENGTH  
04 PAL DB 00H  
05 DATA ENDS  
06 EXTRA SEGMENT  
07 STRING2 DB LEN DUP(?)  
08 EXTRA ENDS  
09 CODE SEGMENT  
10 ASSUME CS: CODE, DS: DATA, ES: EXTRA  
11 START:  
12 MOV AX, DATA  
13 MOV DS, AX  
14 MOV AX, EXTRA  
15 MOV ES, AX  
16 LEA SI, STRING1  
17 MOU BX, LEN-1  
18 LEA DI, STRING2 + BX  
19 MOU CX, LEN  
20 BACK:CLD  
21 LODSB  
22 STD  
23 STOSB  
24 LOOP BACK  
25 LEA SI, STRING1  
26 LEA DI, STRING2  
27 MOU CX, LEN  
28 CLD  
29 REPE CMPSB  
30 JNZ SKIP  
31 INC PAL  
32 SKIP: INT 03H  
33 CODE ENDS  
34 END START  
35  
36 ;NAME: KONDROLLA DINESH REDDY  
37 ;ROLL.NO: 2020BTechCSE040
```



## RESULT:



## **EXPERIMENT – 13:**

**AIM:** Write an ALP to Convert a Decimal Number into Hexadecimal. Assume the Decimal Number is stored at 24000H. Store the result at 24001H.

**CODE:**