1 – calculator
2 – Average
3 – Number of 1
7 – Factorial
10 – Prime
12 – Tables
14 – Fibonacci
17 – Compare
19 – Length
21 – Number of Vowels
23 – Reverse
29 – Conversions
34 – Square root
39 – MASM
42 – KEYPAD

Microprocessor and Interfacing Lab Experiment1

Name: Om Ashish Mishra
Registration No.: 16BCE0789
Slot: B2
Date: 11/12/2017

Write an ALP

1. To perform basic arithmetic operations using 8 bit and 16 bit number (Add, SUB, MUL, and DIV).

16-Bit

Addition

```
mov ax, 5000h
mov ds, ax
mov ax, 23h
mov bx, 10h
add ax, bx
hlt
```

Subtraction

```
mov ax, 5000h
mov ds, ax
mov ax, 23h
mov bx, 10h
sub ax, bx
hlt
```

Multiplication

```
mov ax, 5000h
mov ds, ax
mov ax, 200h
mov bx, 4
mul bx
ret
```

Division

```
mov ax, 5000h
mov ds, ax
mov ax, 0020h
mov bx, 4
div bx
ret
```

8-Bit

Addition

```
mov al, 5000h
mov ah, al
mov al, 23h
mov bl, 10h
add ah, bl
hlt
```

Subtraction

```
mov al, 5000h
mov ah, al
mov al, 23h
mov bl, 10h
sub ah, bl
hlt
```

Multiplication

```
mov al, 5000h
mov ah, al
mov al, 200h
mov bl, 4
mul bl
ret
```

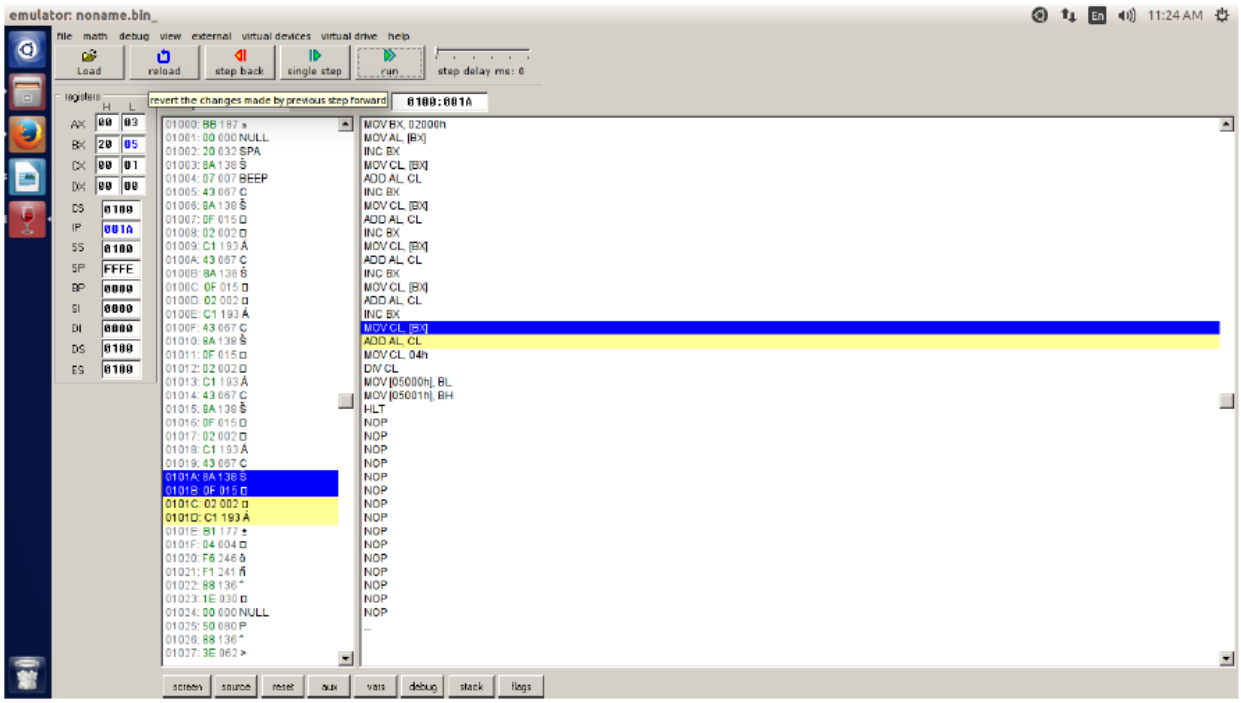
Division

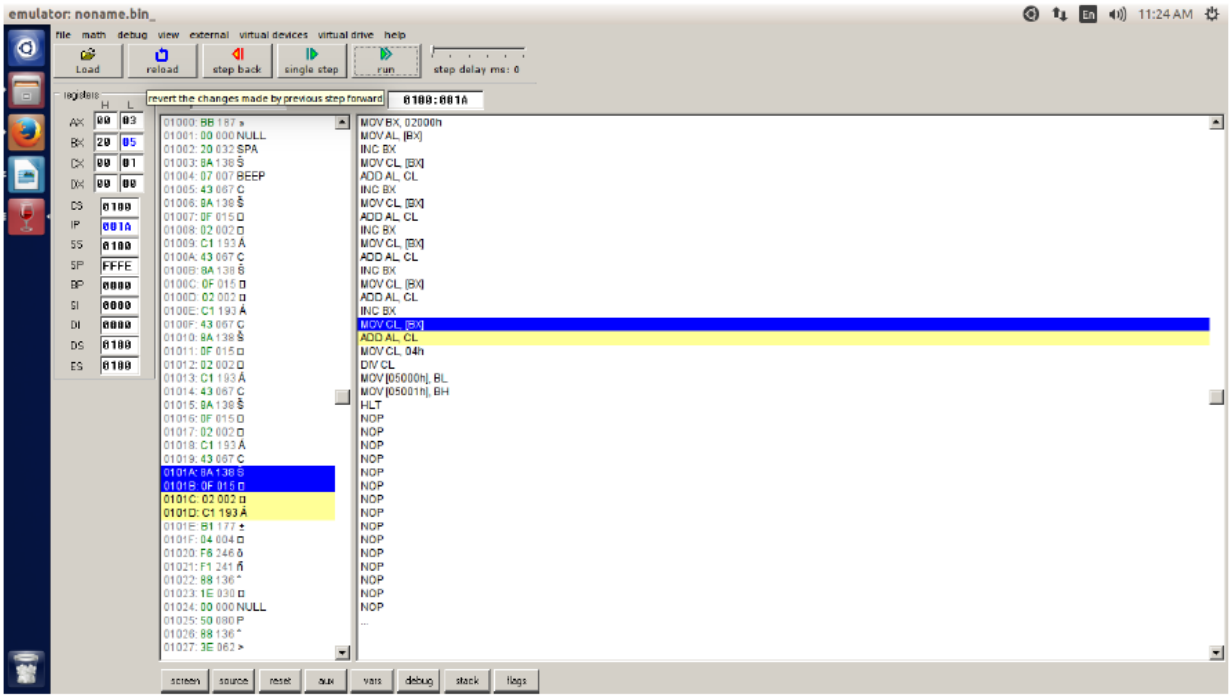
```
mov al, 5000h
mov ah, al
mov al, 0020h
mov bl, 4
div bl
ret
```

2. Write an ALP to perform the sum and average of 5 numbers stored in the address location 2000h onwards and put the result in memory locations starting from 5000h onwards.

```
mov bx, 02000h
mov al,[bx]
inc bx
mov cl,[bx]
add al,cl
inc bx
mov dl,[bx]
add al,cl
inc bx
```

mov dl,[bx]
add al,dl
mov cl,04h
div cl
mov [5000h],al
mov [5001h],ah
hlt
Test Cases : Input Value and the result:
Input Value:

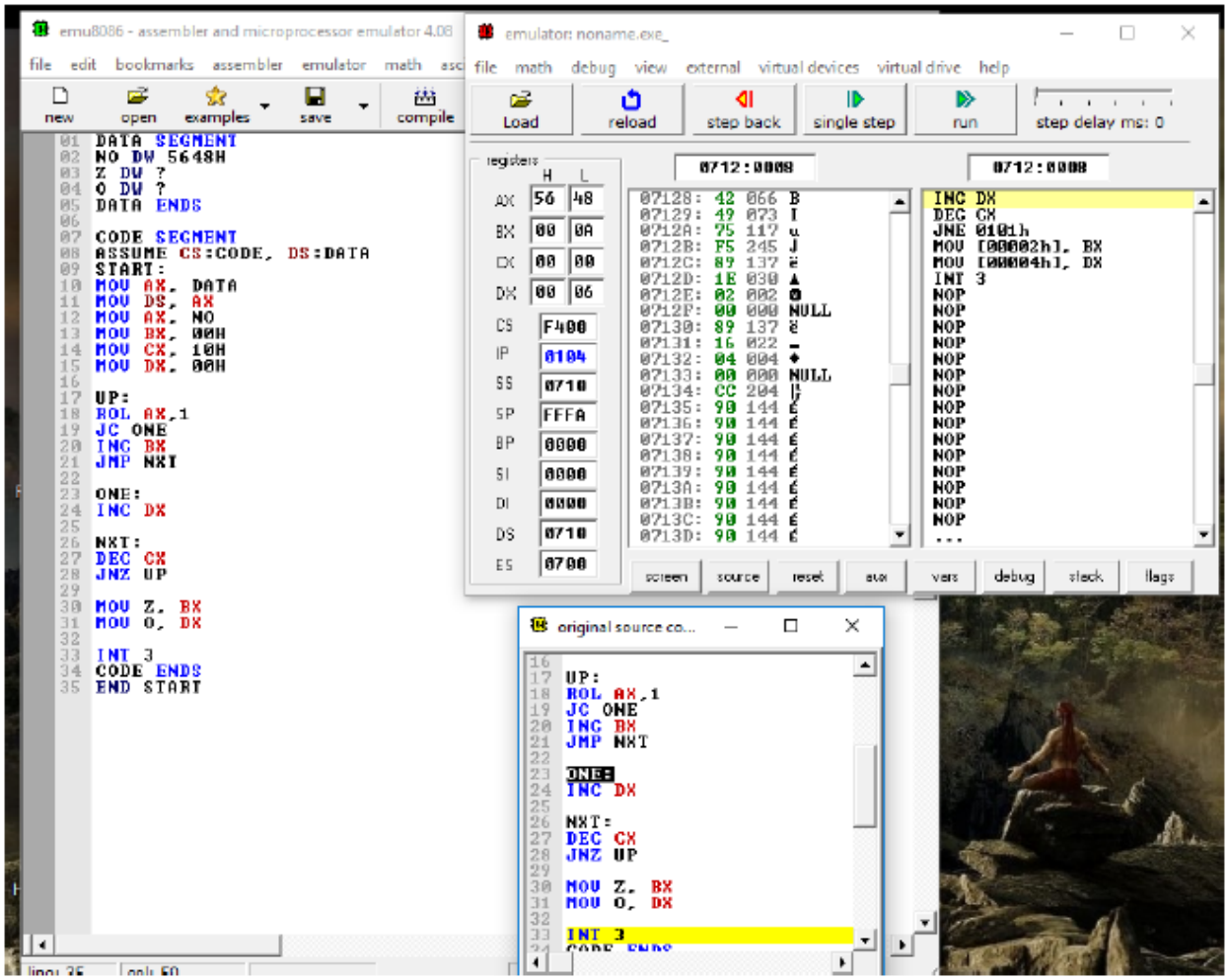




3. Write a program to count the number of 1's in a given byte located in AL register..(Hint Use Rotate instruction : Eg : Input(AL) : 0A7 h Output : 5))

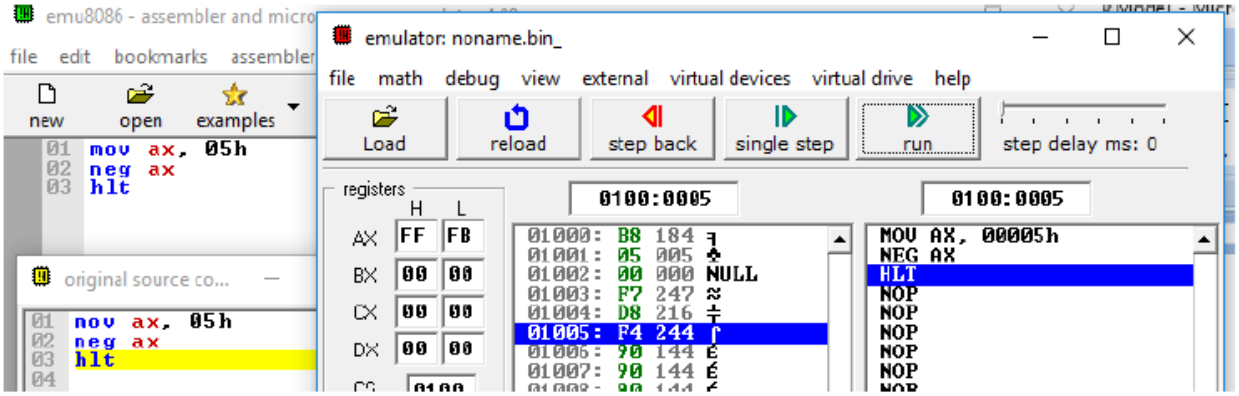
```
DATA SEGMENT
NO DW 5648H
Z DW ?
O DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
MOV AX, DATA
MOV DS, AX
MOV AX, NO
MOV BX, 00H
MOV CX, 10H
MOV DX, 00H
UP:
ROL AX,1
JC ONE
INC BX
JMP NXT
ONE:
INC DX
NXT:
DEC CX
JNZ UP
```

```
MOV Z, BX
MOV O, DX
INT 3
CODE ENDS
END START
```

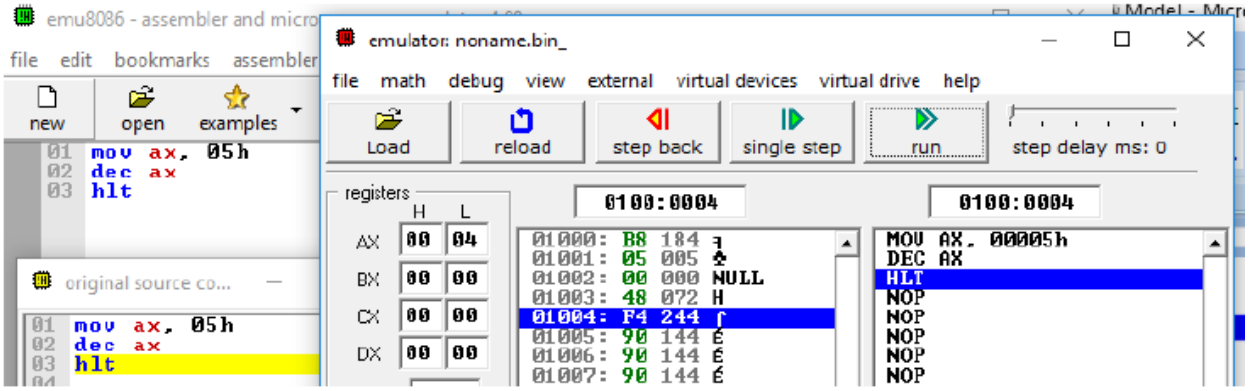


4. Analyse Any five instructions – NEG,DEC, AAM ,IMUL SAR,SL,SR(take any input number and show the output)

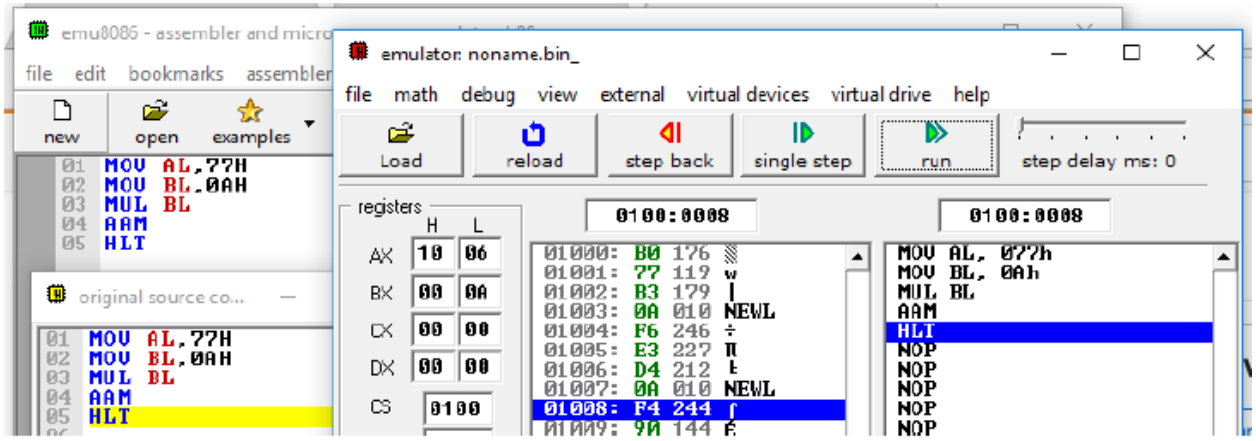
NEG: Creates 2's complement of original value (i.e., forms the negative of a given value).



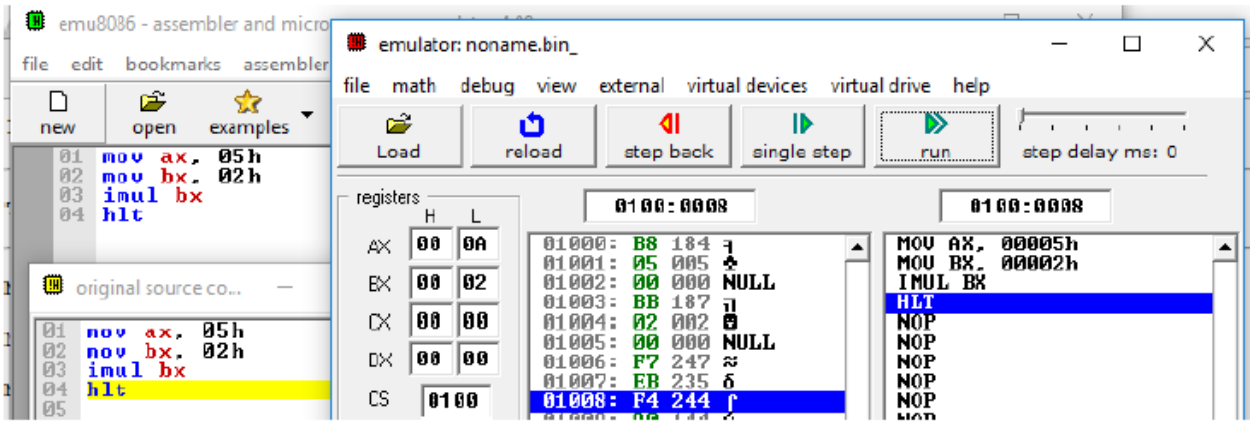
DEC: It is used to decrease the value by one.



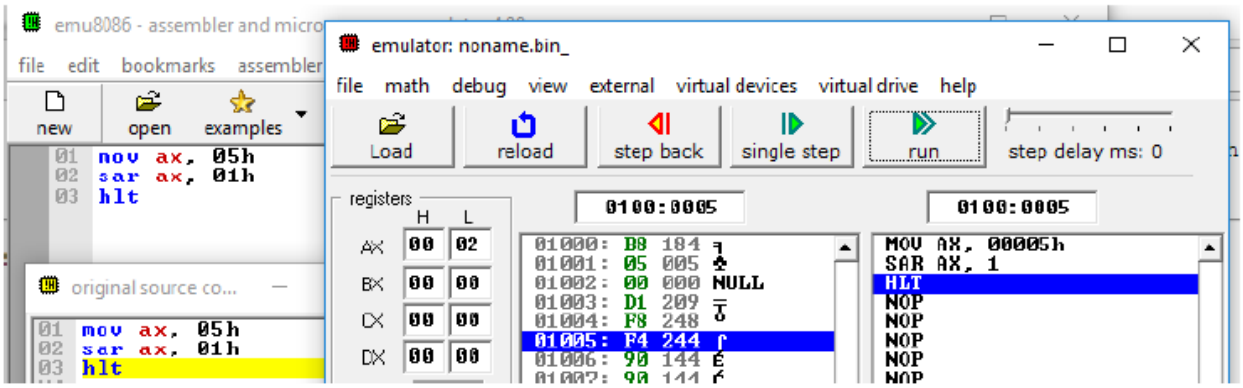
AAM: Adjusts the result of the multiplication of two unpacked BCD values to create a pair of unpacked (base 10) BCD values. The AX register is the implied source and destination operand for this instruction. The AAM instruction is only useful when it follows an MUL instruction that multiplies (binary multiplication) two unpacked BCD values and stores a word result in the AX register. The AAM instruction then adjusts the contents of the AX register to contain the correct 2-digit unpacked (base 10) BCD result.



IMUL: This is used for signed multiplication



SAR: shifts to the right and preserves the sign bit for positive or negative numbers. I.e., you should use SAR instead of SHR when you are using signed numbers in a program.



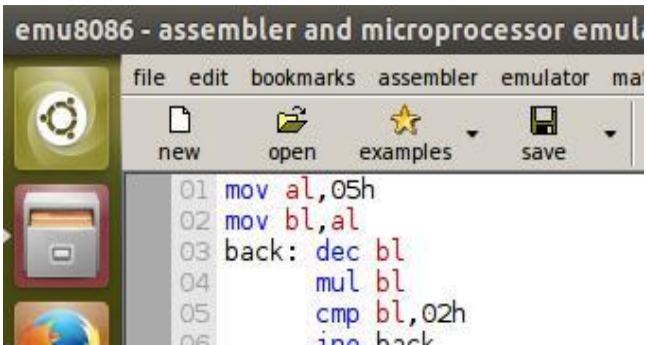
1. Write an ALP to find the factorial of a number with and without using loop instruction

LOOP

Factorial

Write a program to find the factorial of a number with /without using loop instruction.

Without Loop



Aim: Factorial without loop

Algorithm:
Step 1: First we store the value of the 5 in al.

Step 2: Then we store the value of al in bl which will calculate the factorail Step 3: Then we decrease the value of bl.

Step 4: The multiplication takes place between bl prevoius value and bl.

Step 5: Compare the new value with 02h

Step 6: if not euql then we back to dec bl and to multiply with bl. Step 7: else we got to hlt

Step 8: We get the result.

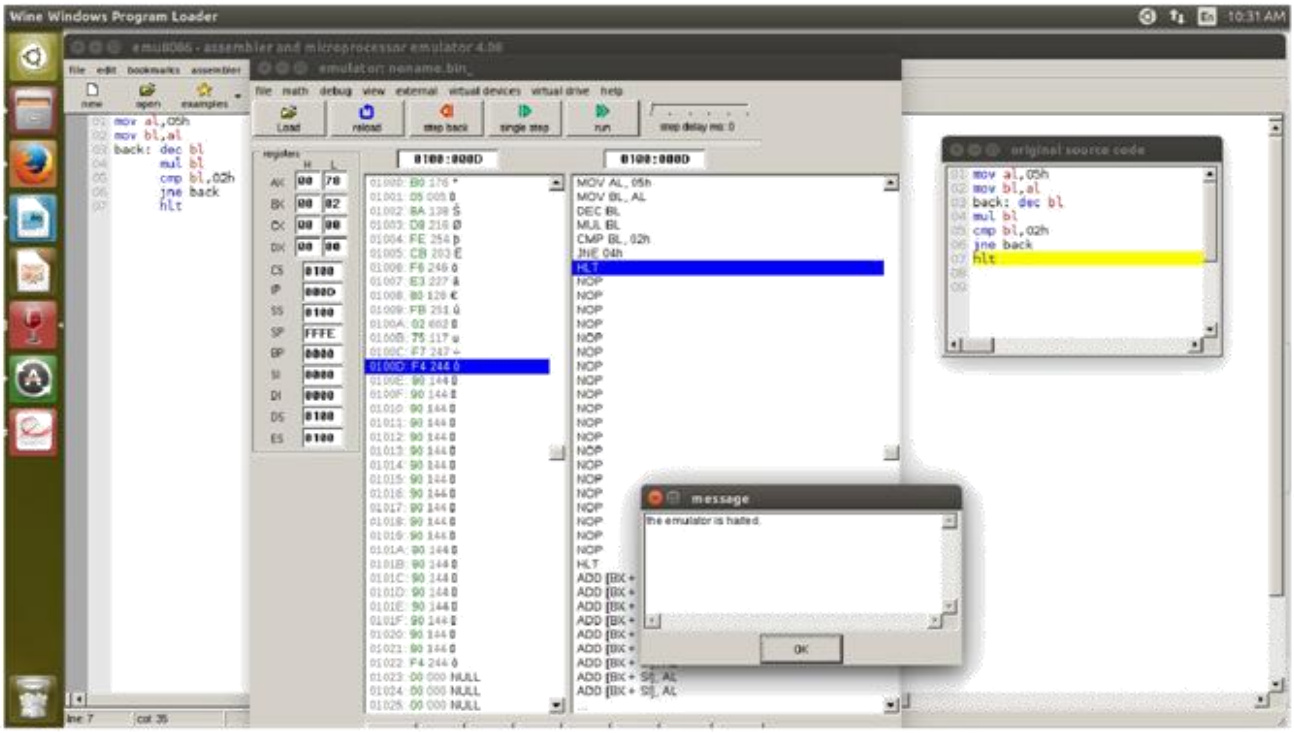
Memory Address	Instruction	Hex code
1000 1001	Mov al,05h	B0,05
1002 1003	Mov bl,al	8A,D8
1004 1005	Back: dec bl	FE,CB
1006 1007	Mul bl	F6,E3
1008 1009 100A	Cmp bl,02h	80,FB,02
100B 100C	Jne back	75,F7
100D	hlt	F4

Sample Input: 5h

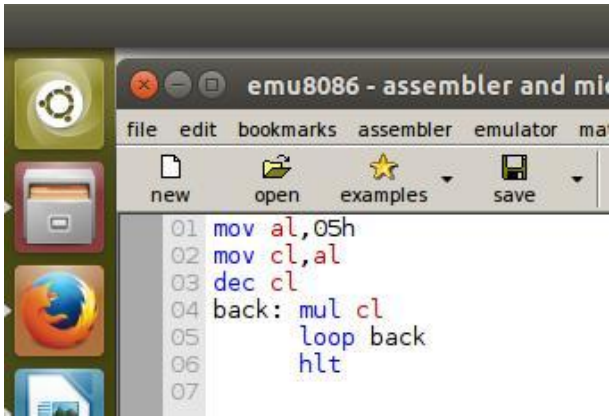
Sample Output: 78h

Result:

The result of factorial is 78h.



With Loop



Aim: Factorial with loop

Algorithm:

Step 1: First we store the value of the 5 in al.

Step 2: Then we store the value of al in cl which will calculate the factorial Step 3: Then we decrease the value of cl.

Step 4: The multiplication takes place between cl prevoius value and cl.

Step 5: We continue to loop in 01h

Step 8: We get the result.

Memory Address	Instruction	Hex code
1000 1001	Mov al,05h	B0,05
1002 1003	Mov cl,al	8A,C8
1004 1005	dec cl	FE,C9
1006 1007	Back: mul cl	F6,E1
1008 1009	Loop back	E2,FC
0110A	hlt	F4

Sample Input: 5

Sample Output: 78h

The Result:

The result of factorial is 78h.

2. Write ALP to Check whether the given number is prime or not.

Aim: Prime Number or not

Algorithm:
Step 1: First we ask the user to choose a number

Step 2: Then we store the data in data segement which is transferred it to accumulator in code segment

Step 3: Then we display the number.

Step 4: Then we run a loop till of the number to check whether it is division by the any of the numbers in the loop

Step 5: If it is divisible it is printed as not a print number

Step 6: Else it is printed as a prime number.

Step 7: We got to hlt

Step 8: We get the result.

Memory Address	Instructions	Hex Code
1000 1001	.data	B0,05
1002 1003	AHSAN DB "ENTER NUMBER BETWEEN 0 & 9 ::: $"	8A,C8
1004 1005	AHSAN1 DB 0DH,0AH,"IT IS PRIME NUMBER $"	FE,C9
1006 1007	AHSAN2 DB 0DH,0AH,"IT IS NOT PRIME NUMBER $"	F6,E1
1008 1009 100A	.CODE	E2,FC
100B 100C	LEA DX,AHSAN	F4, B4
100D	MOV AH,09H	B0
100E 100F	INT 21H	8A,C8
1010 1011	MOV AH,01H	FE,C9
1012 1013	INT 21H	F6,E1
1014 1015	SUB AL,30H	E2,FC
1016 1017 1018	MOV BL,AL	F4,1E,A8
1019 101A	CMP BL,02H	B0,05
101B	JE PRIME	8A
101C 101D	CMP BL,03H	FE,C9
101E	JE PRIME	F6
101F 1020	CMP BL,05H	E2,FC
1021	JE PRIME	F4
1022 1023	CMP BL,07H	B0,05
1024	JE PRIME	8A
1025	JMP NOTPRIME	FE
1026 1027	PRIME:	F6,E1
1026 1027	LEA DX,AHSAN1	F6,E1
1028 1029	MOV AH,09H	F4,D5
102A 102B	INT 21H	B0,05
102C	JMP PEND	8A
102D 102E	NOTPRIME:	FE,C9
102D 102E	LEA DX,AHSAN2	FE,C9
102F 1030	MOV AH,09H	E2,FC
1031	INT 21H	F4
1032	JMP PEND	B0
1033 1034	PEND:	8A,C8
1033 1034	MOV AH,4CH	FE,C9

1035 1036	INT 21H	F6,E1
1037	RET	E2

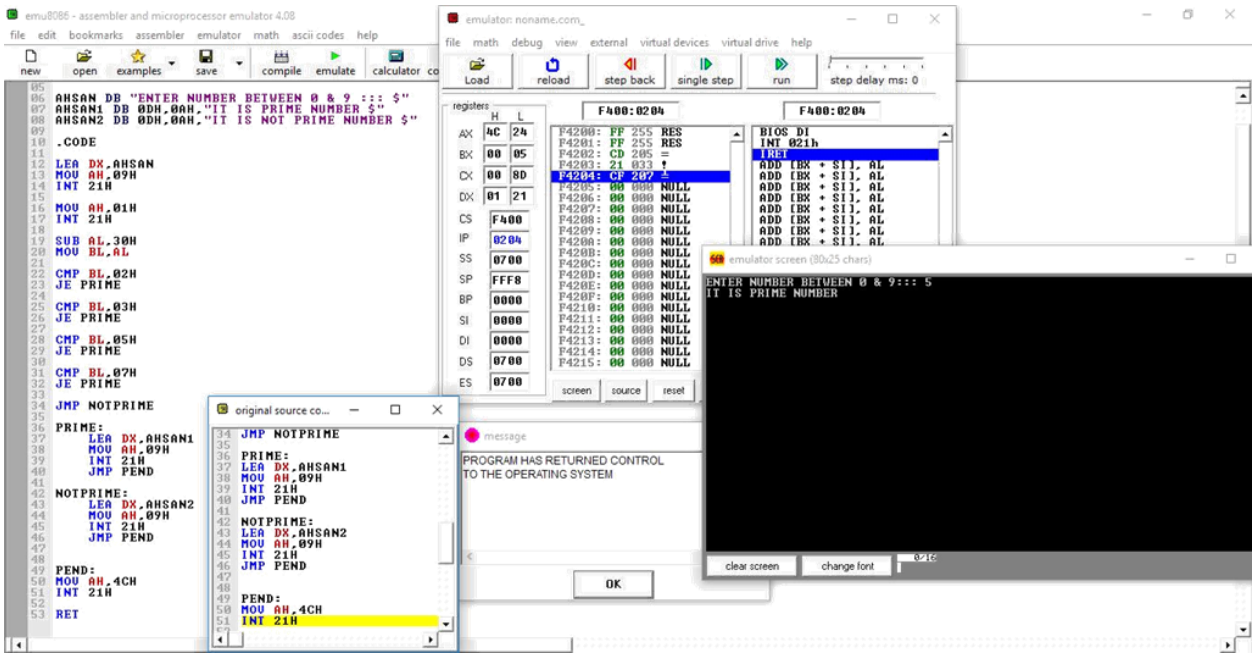
Sample Input: 5

Sample Output: Prime

Result:

Enter a number between 0 and 9: 5

It is a prime number.



3. Write ALP to print the multiplication table for number 07.

Aim: Multiplication of table 7

Algorithm:
Step 1: First we store the value of the 7 in al.

Step 2: Then we store value of al in bl

Step 3: Then we store value of 01h in cl which is the counter.

Step 4: We call si(stack Initiator) to store the value of address 2000h.

Step 5: Now the looping starts and we multiply cl's value with bl.

Step 6: increase the value of the address location

Step 7: Refresh the value of the al from

bl Step 8: increase the counter value

Step 9: compare for equal to 10h

Step 10: if yes we hlt and get the result.

Step 11: else we go back into the loop

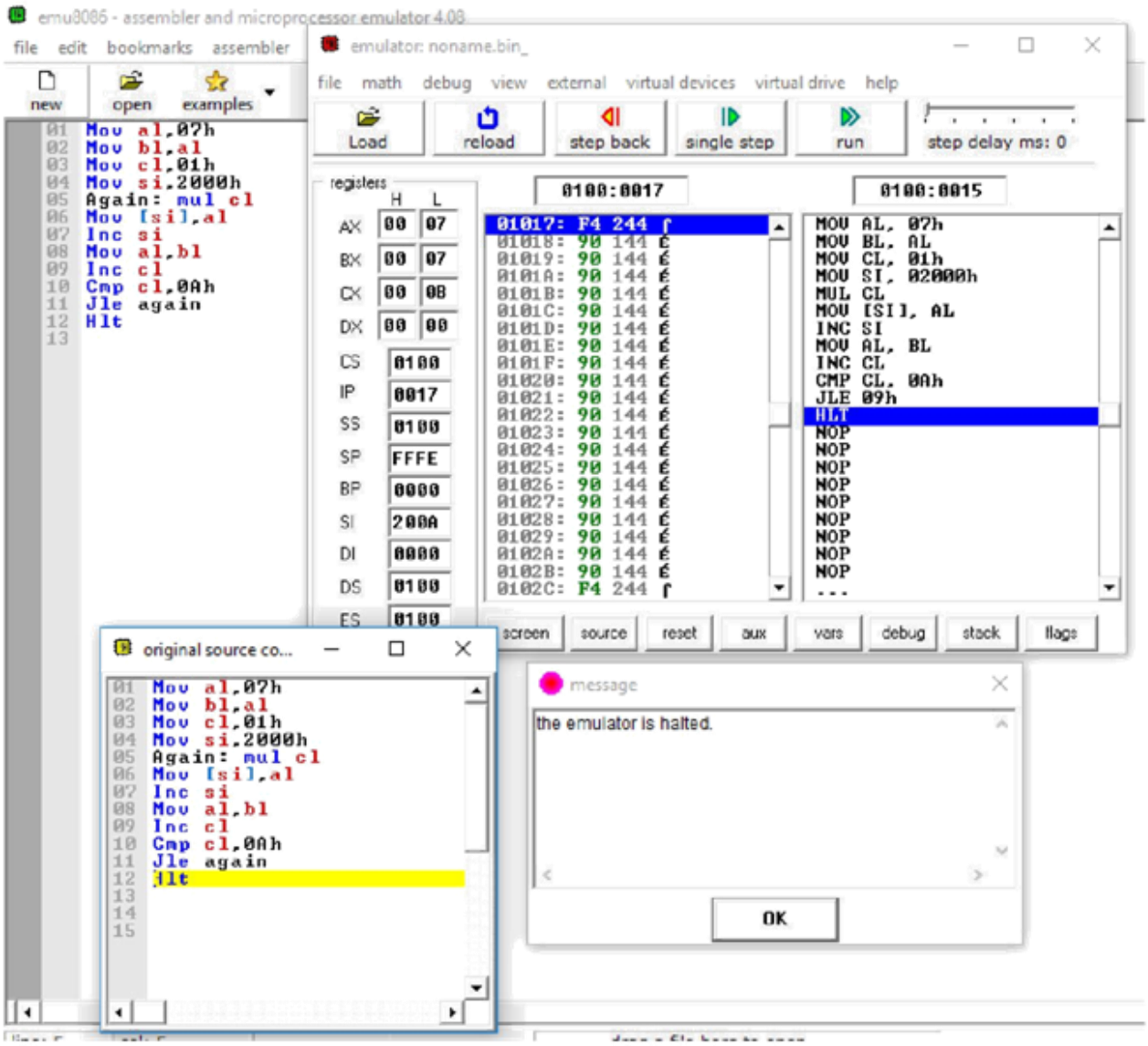
Memory Address	Instruction	Hex code
1000 1001	Mov al,07h	B0,05
1002 1003	Mov bl,al	8A,D8
1004 1005	Mov cl,01h	B1,01
1006 1007 1008	Mov si,2000h	BE,00,20
1009 100A	Agian: mul cl	F6,E1
0100B 0100C	Mov [si],al	88,04
0100D	Inc si	46
0100E 0100F	Mov al,bl	8A,C3
01010 01011	Inc cl	FE,C1
01012 01013 01014	Cmp cl,0Ah	80,F9,0A
01015 01016	Jle, again	7E,F2
01017	Hlt	F4

Sample Input: 7

Sample Output: Table of 7

Result:

The table represents multiplication of 7 from 1 to 10.



4. Write ALP to generate the Fibonacci series for N terms.

Aim: Fibonacci series

Algorithm:

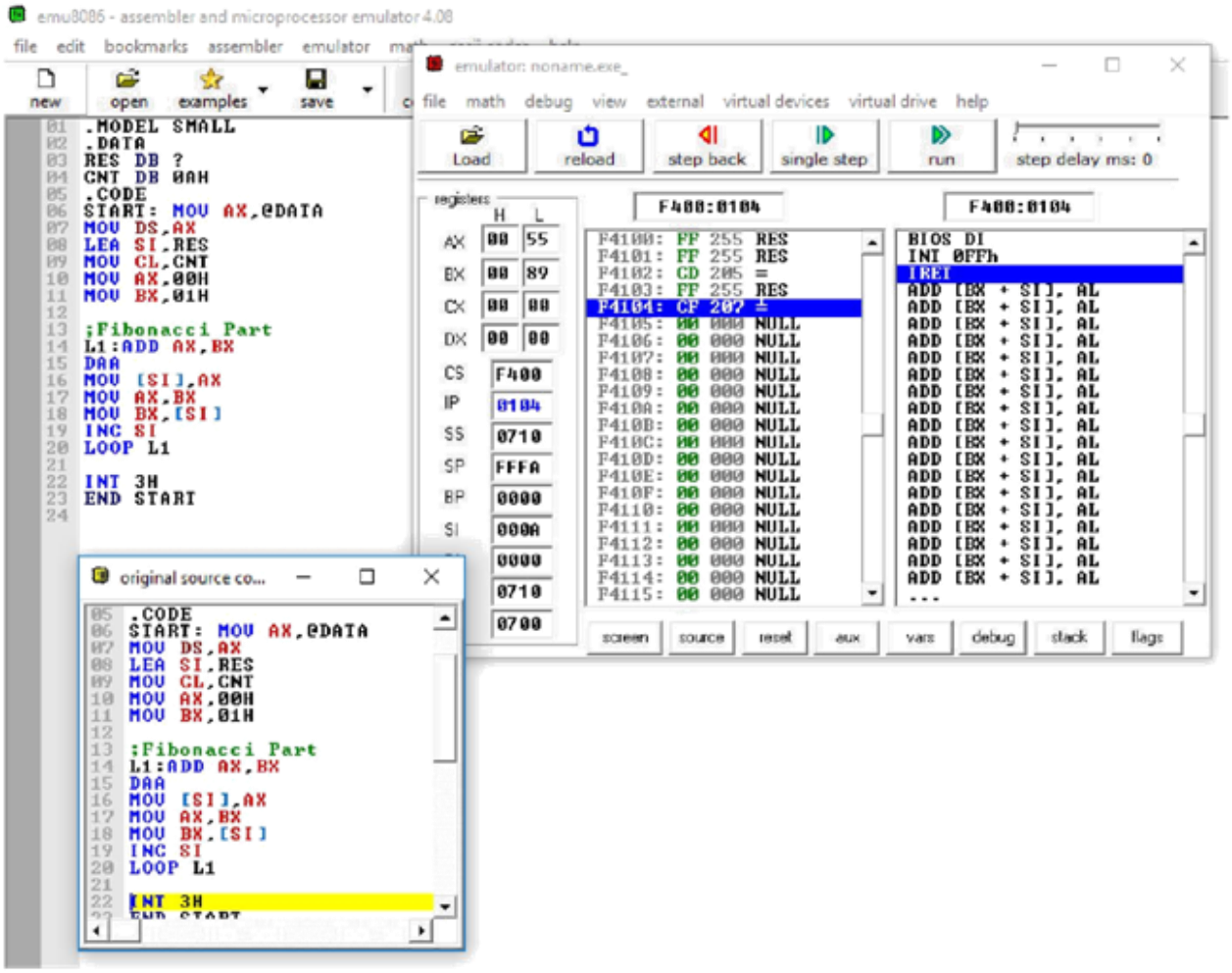
Step 1: First we store the value of 1 and 0 in data segment and counter register is created here its value is 10.

Step 2: Then we store it in accumulator in code segment and do addition in loop for Fibonacci.

Step 3: First we run the loop and keep on adding and swapping the positions of numbers till we reach counter as 0.

Step 4: Show the output and stop.

Memory Address	Instructions	Hex Code
1000 1001	.MODEL SMALL	B0,05
1002 1003	.DATA	8A,D8
1004 1005	RES DB ?	B1,01
1006 1007 1008	CNT DB 0AH	BE,00,20
1009 100A	.CODE	F6,E1
0100B 0100C 0100D	START: MOV AX,@DATA	88,04
0100E 0100F	MOV DS,AX	46,C8
01010 01011	LEA SI,RES	8A,C3
01012 01013	MOV CL,CNT	FE,C1
01014 01015	MOV AX,00H	80,F9
01016 01017	MOV BX,01H	7E,F2
01018 01019	L1:ADD AX,BX	F4,C7
0101A	DAA	B0
0101B 0101C	MOV [SI],AX	8A,D8
0101D 0101E	MOV AX,BX	B1,01
0101F 01020	MOV BX,[SI]	BE,00
01021	INC SI	F6
01022	LOOP L1	88
01023	INT 3H	4B
01024	END START	8A



Sample Input: 10

Sample Output: 55h

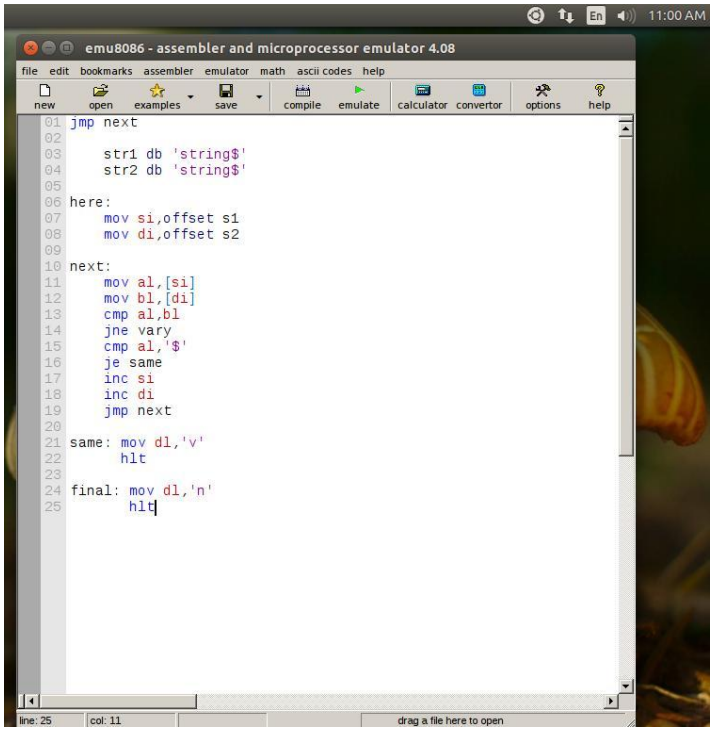
Result:

The sum of Fibonacci is 55h (88 in decimal).

Write an ALP to perform the following string operations length, reverse and compare.

This question I have divided into 3 parts.

2. Compare



Aim: Compare two strings

Algorithm:

Step 1: First we store the value f two strings one in si and other in di using offset.

Step 2: Then we compare them using cmp instruction.

Step 3: If it varies we go to vary and stop else we continue.

Step 4: Then we check whether we reached the end if yes we go to same and stop.

Step 5: Inorder go back into the loop we use increase of si and di.

Step 6: We get the result same or not.

Memory Address	Instruction	Hex code
1000 1001	Jmp next	B0,05
1002 1003	Str1 db 'string$'	8A,D8
1004 1005	Str2 db 'string$'	FE,CB
1006 1007	Here: mov si,offset s1	F6,E3
1008 1009 100A	Mov di, offset s2	80,FB,02
100B 100C	Next: mov al,[si]	75,F7
100D	Mov bl, [di]	F4
1016 1017	Cmp al,bl	74 1A
1018 1019	Jne vary	3C 61
101A 101B	Cmp al,'$'	74 0B
101C 101D	Je same	3C 65
101E 101F	Inc si	74 0E
1020 1021	Inc di	3C 69
1022 1023	Jmp next	74 0A
1024 1025	Same: mov dl, 'v'	3C 6F
1026 1027	Hlt	74 06
1028 1029	Final: mov dl,'n'	3C 75
102A 102B	Hlt	74 02

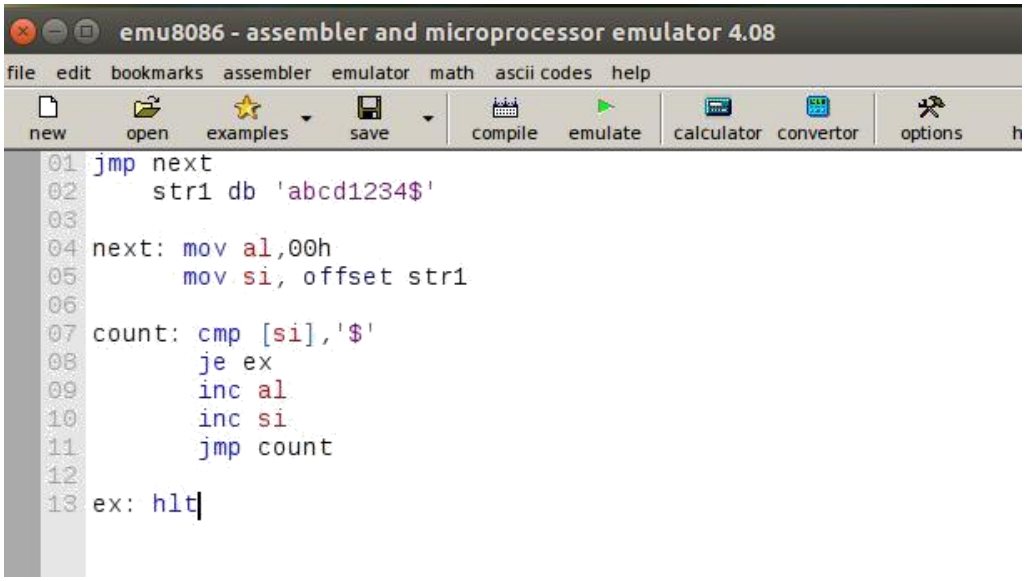
Sample Input: Ram,Ram

Sample Output: Equal

Result:

The result is both the strings are equal.

(ii) Length



Aim: Length of the

string Algorithm:

Step 1: First we store the value of the string

Step 2: Then we run a loop check if we have reached the end.

Step 3: In the process the value of counter keeps on increasing and thus the program ends

Step 4: We get the result.

Memory Address	Instruction	Hex code
1000 1001	Jmp next	B0,05
1002 1003	Str1 db 'abcd1234$'	8A,D8
1004 1005	Next: mov al,00h	FE,CB

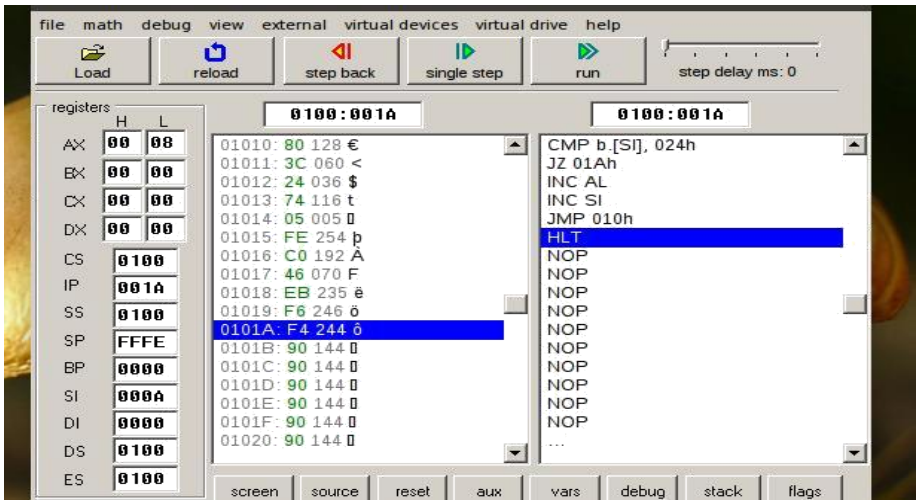
1006 1007	mov si,offset s1	F6,E3
1008 1009 100A	Count: cmp [si], '$'	80,FB,02
100B 100C	Je ex	75,F7
100D	Inc al	F4
1016 1017	Inc si	74 1A
1018 1019	Jmp count	3C 61
101A 101B	Ex: hlt	74 0B

Sample Input: God is Great!!

Sample Output: Dh

Result:

The length of the string is Dh.(14 latters long in decimal).



2. To Count the number of vowels in a given string.

```
01 jmp next
02
03     str db 'Education$'
04     lea si,str
05     mov bl,000h
06
07 next:
08     mov al,[si]
09     inc si
10     cmp al,'$'
11     je final
12     cmp al,'a'
13     je count
14      cmp al,'e'
15     je count
16      cmp al,'i'
17     je count
18      cmp al,'o'
19     je count
20      cmp al,'u'
21     je count
22     jmp next
23
24 count: inc bl
25        jmp next
26
27 final: hlt
```

Aim: Count number of vowels.

Algorithm:

Step 1: First we take a string and store it (here the string is Education)

Step 2: Then we change the value of bl as 0h as we are going to use it for counter.

Step 3: Then we run a loop where we check if it is a vowel ('a','e';'i','o','u').

Step 4: If vowel the value of bl is increased else we check the next element by increasing si and going back to the loop.

Step 5: On every count the value of count increases Step

6: Thus we get the value no of vowels and we stop.

Memory Address	Instruction	Hex code
1000 1001	Jmp next	EB,0F
1002 1003	Str db 'education$'	8A,D8

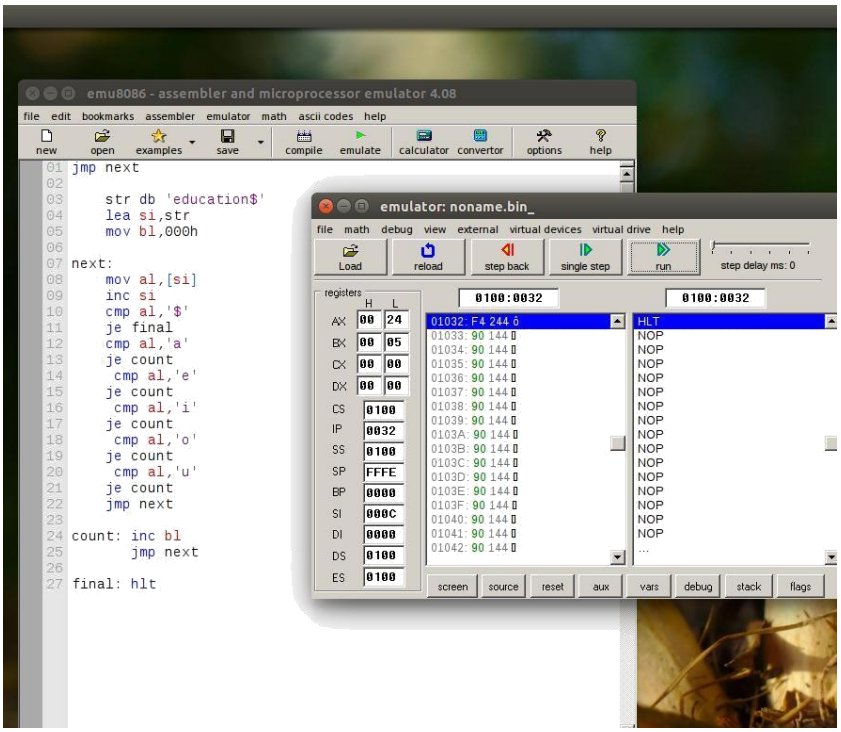
1004 1005	Lea si,str	FE,CB
1006 1007	Mov bl,000h	8A,4F
1011 1012	Next: mov al,[si]	8A,4f
1013	Inc si	46
1014 1015	Cmp al,'$'	3C,24
1016 1017	Je final	74 1A
1018 1019	Cmp al,'a'	3C 61
101A 101B	Je final	74 0B
101C 101D	Cmp al,'e'	3C 65
101E 101F	Je final	74 0E
1020 1021	Cmp al,'i'	3C 69
1022 1023	Je final	74 0A
1024 1025	Cmp al,'o'	3C 6F
1026 1027	Je final	74 06
1028 1029	Cmp al,'u'	3C 75
102A 102B	Je count	74 02
102C 102D	Jmp next	EF E4
102E 102F	Count: inc bl	EB E3
1030 1031	Jmp next	FE E3
10321033	Final: hlt	F4

Sample Input: Education

Sample Output: 05h

Result:

The result of number of vowels is 05h.



(iii) Reverse

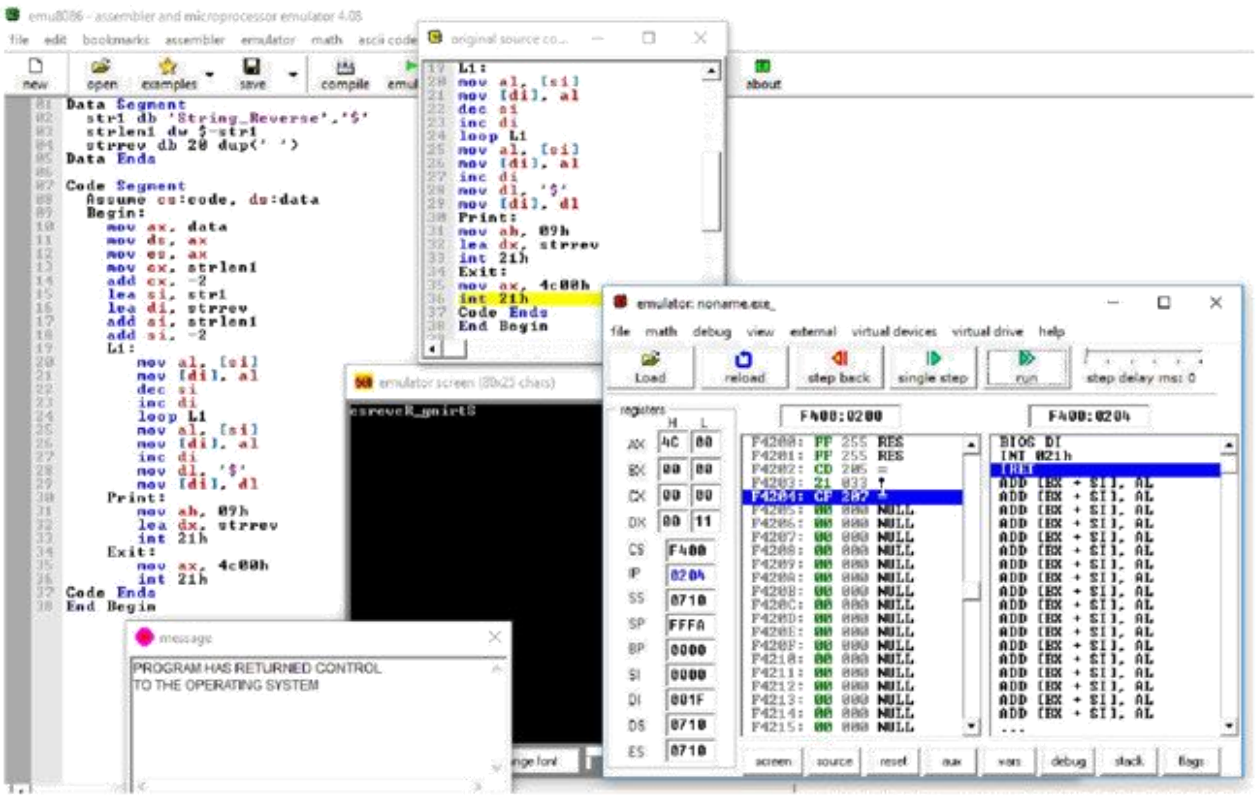
Aim: Reverse the given string

Algorithm:

Step 1: First we get the string in the data segment.
Step 2: Then we store it in accumulator
Step 3: Then we run a loop to check the rotate the text.(this is done by store the string in another string in opposite direction.
Step 4: Then we print the reversed
string Step 5: Halt the code

Memory Address	Instruction	Hex code
1000	Data Segment	B0
1002 1003	str1 db 'String_Reverse','$'	8A,D8
1004 1005	strlen1 dw $-str1	FE,CB
1006 1007	strrev db 20 dup(' ')	F6,E3
1008	Data Ends	80

1009	Code Segment	F4,6E
100A 100B	Assume cs:code, ds:data	8A,D8
100C 100D	Begin:	FE,CB
100C 100D	mov ax, data	FE,CB
100E 100F	mov ds, ax	80,FB
1010 1012	mov es, ax	B0,05
1013 1014	mov cx, strlen1	8A,D8
1015 1016	add cx, -2	FE,CB
1017 1018	lea si, str1	F6,E3
1019 101A	lea di, strrev	80,FB
101B 101C	add si, strlen1	B0,05
101D 101E	add si, -2	8A,D8
101F 1020	L1:	FE,7C
101F 1020	mov al, [si]	FE,7C
1021 1022	mov [di], al	80,FB
1023	dec si	B5
1024	inc di	4A
1025	loop L1	E8
1026 1027	mov al, [si]	F6,E3
1028 1029	mov [di], al	80,FB
102A	inc di	D3
102B 102C	mov dl, '$'	8A,D8
102D 102E	mov [di], dl	FE,CB
102F 1030	Print:	F6,E3
102F 1030	mov ah, 09h	F6,E3
1031 1032	lea dx, strrev	B0,05
1033	int 21h	8A,D8
1034 1035	Exit:	F6,CB
1034 1035	mov ax, 4c00h	F6,CB
1036	int 21h	80
1037	Code Ends	B9
1038	End Begin	8B



Sample Input: we are here

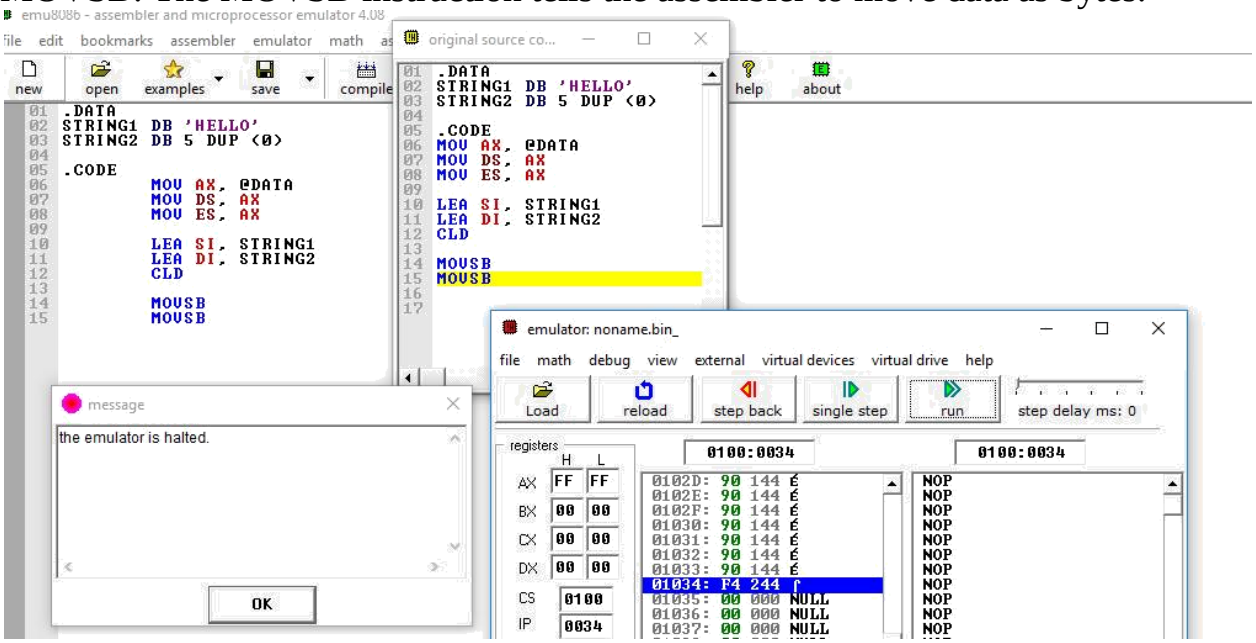
Sample Output: ereh era ew

Result:

The reverse string is given of the input.

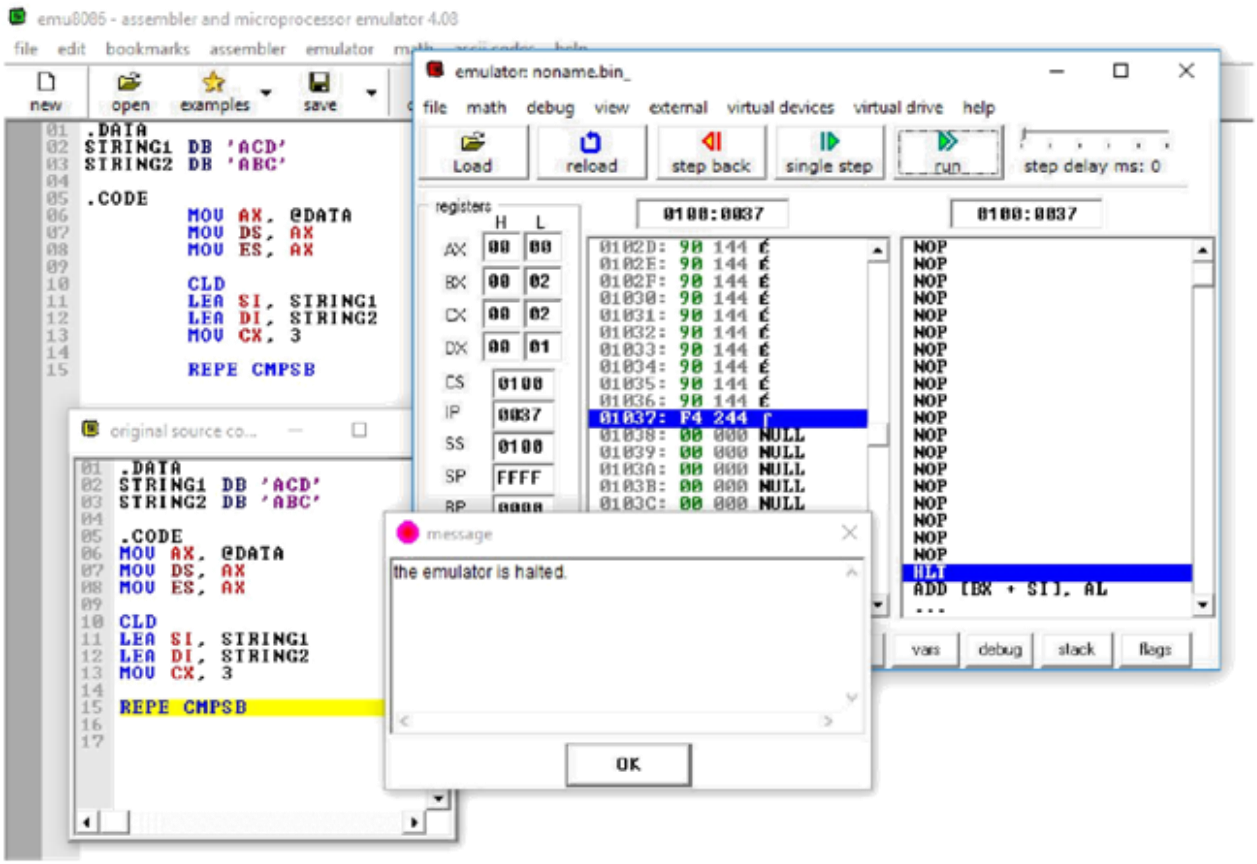
3. Analyse the string Instruction MOVSB, CMPS, SCAS etc.

MOVSB: The MOVSB instruction tells the assembler to move data as bytes.



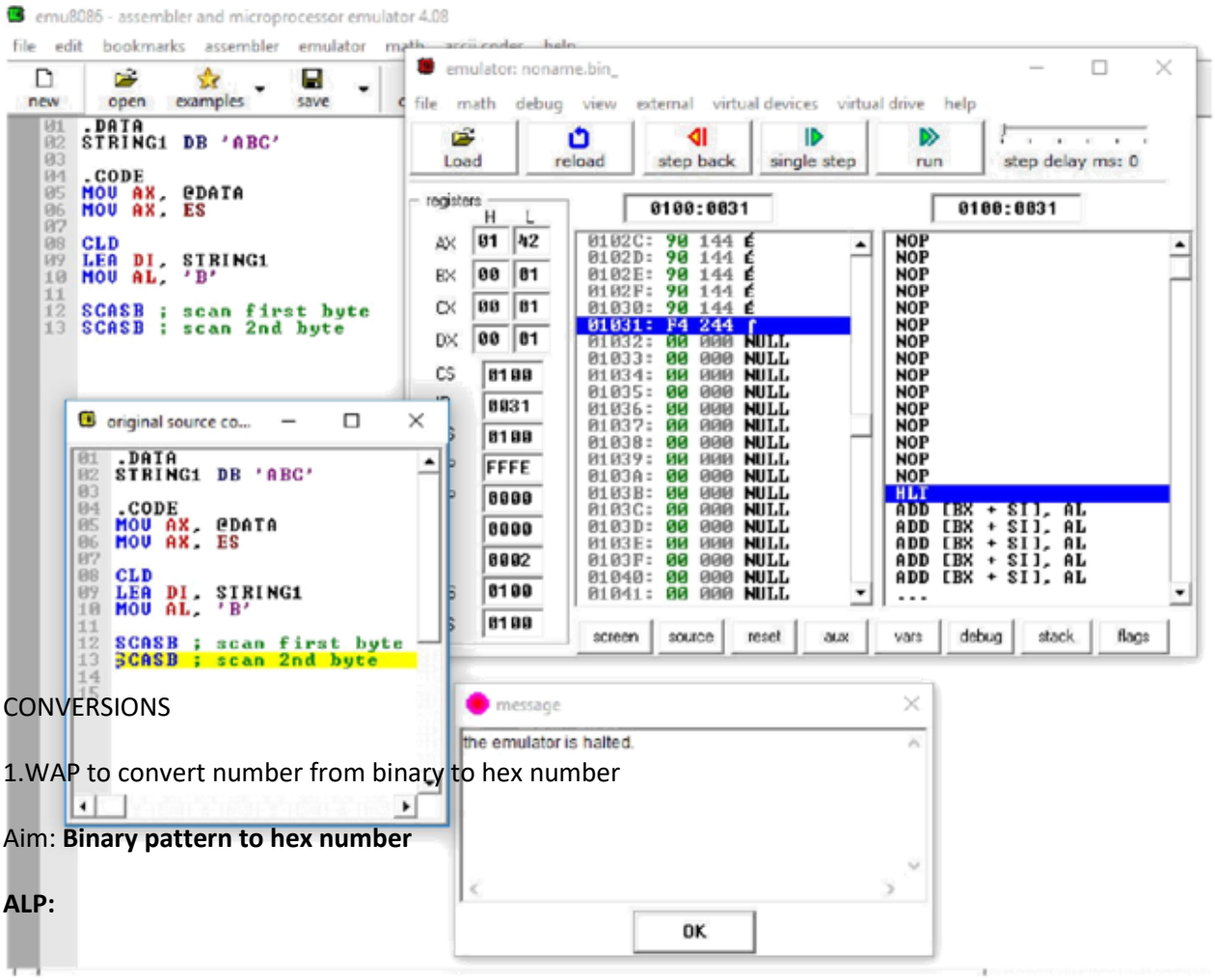
CMPS: This is used to compare the strings, words and double words

CMPSB - compares BYTE at ES:DI with BYTE at DS:SI and sets flags.
CMPSW - compares WORD at ES:DI with WORD at DS:SI and sets flags.
CMPSD - compares DOUBLE WORD at ES:DI with WORD at DS:SI and sets flags.



SCAS: This instructyion is used to do various kind of comparisons

SCASB - compares BYTE at ES:DI with AL and sets flags according to result.
SCASW - compares WORD at ES:DI with AX and sets flags.
SCASD - compares DOUBLE WORD at ES:DI with EAX and sets flags.

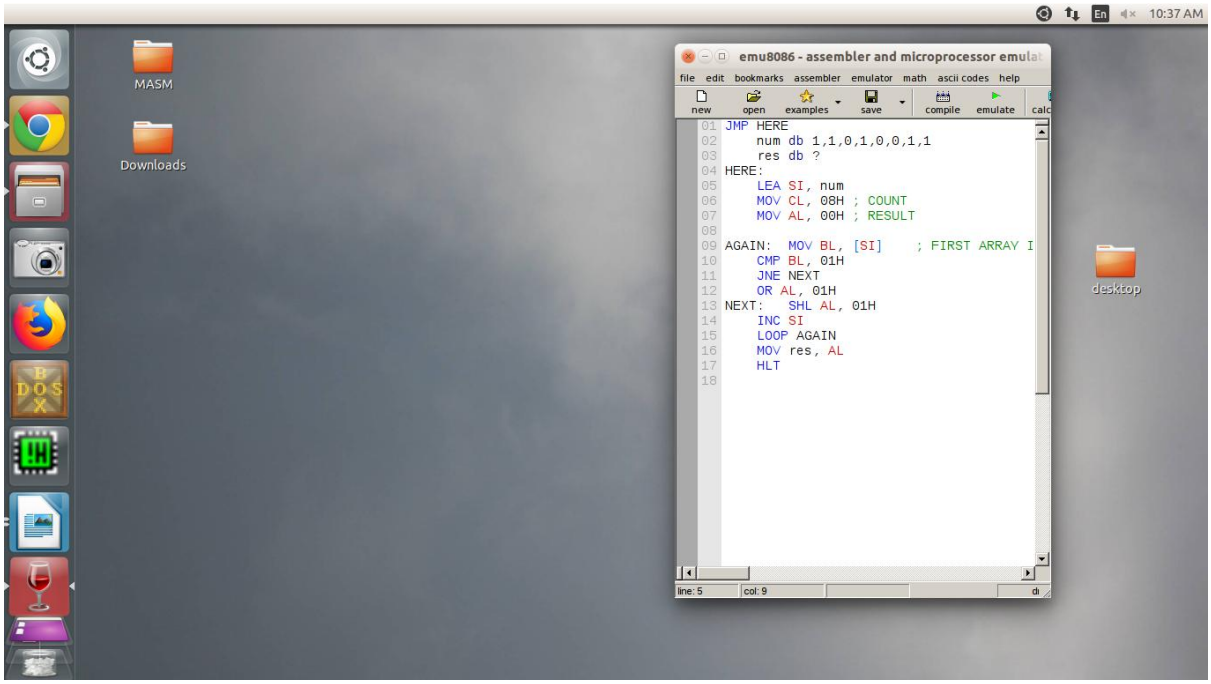


CONVERSIONS

1.WAP to convert number from binary to hex number

Aim: **Binary pattern to hex number**

ALP:




```
JMP HERE
        num db 1,1,0,1,0,0,1,1
        res db ?
HERE:
        LEA SI, num
        MOV CL, 08H    ; COUNT
        MOV AL, 00H    ; RESULT

AGAIN: MOV BL, [SI]    ; FIRST ARRAY ITEM
        CMP BL, 01H
        JNE NEXT
        OR AL, 01H
NEXT:   SHL AL, 01H
        INC SI
```

LOOP AGAIN

MOV res, AL

HLT

Algorithm:

Step 1: First we take a numbers in an array and store it in num

Step 2: Then we load the source register with num

Step 3: Then we initialize counter for calulation and accumulator for result

Step 4: Then we run a loop for using the array elements to compare and check for one.

Step 5: If one we move to next loop, shift left and increase si by one and if zero we store it in accumulator

Step 6: After again loop ends we display the result.

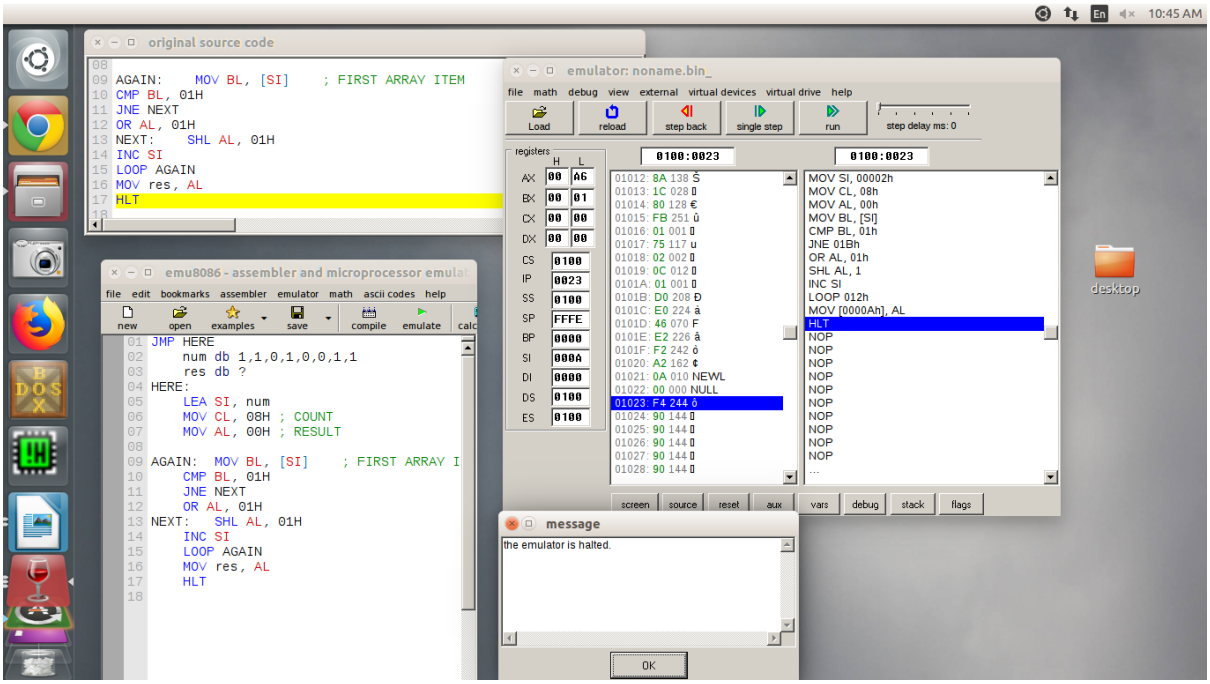
Step 7: We halt the program.

Memory Address	Instruction	Hex code
1000 1001	Jmp HERE	EB,09
100B 100C 100D	HERE: LEA SI, num	BE 02 00
100E 100F	MOV CL,08H	B1 08
1010 1011	MOV AL,00H	B0 00
1012 1013	AGAIN: MOV BL,[SI]	8A 1C
1014 1015 1016	CMP BL,01H	80 FB 01
1017 1018	JNE NEXT	75 02

1019 101A	OR AL,01H	0C 01
101B 101C	NEXT: SHL AL,01H	D0 E0
101D	INC SI	46
101E 101F	LOOP AGAIN	E2 F2
1020 1021 1022	MOV res,AL	A2 0A 00
1023	HLT	F4

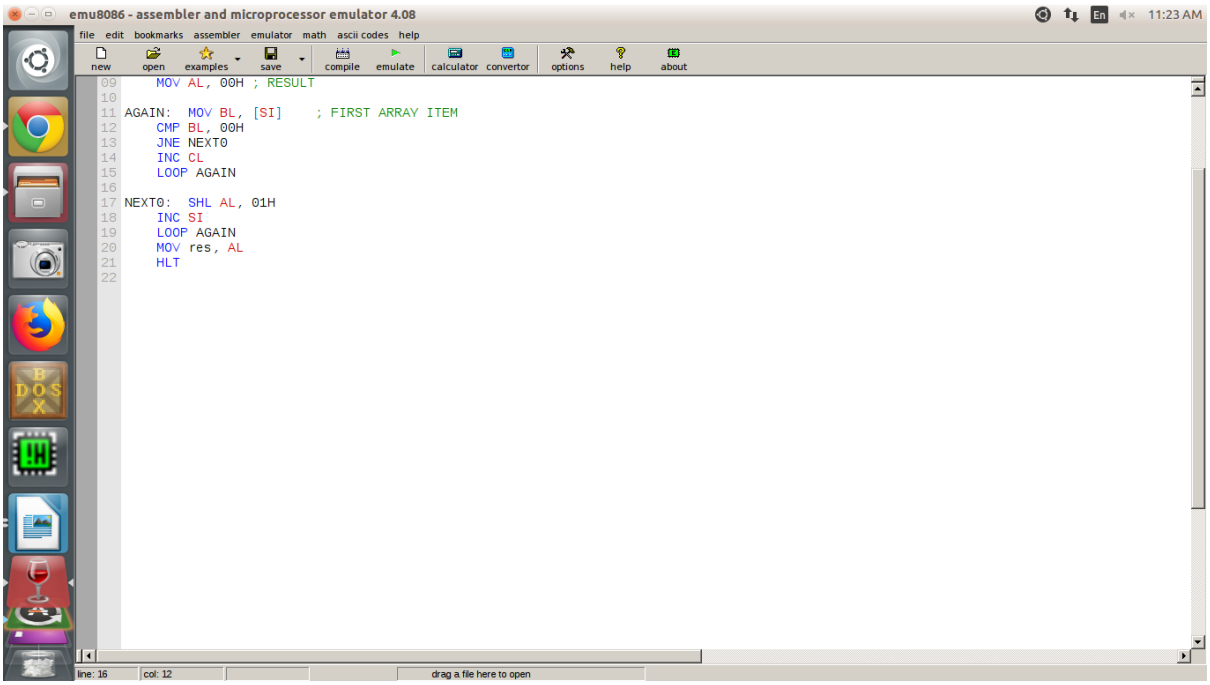
Result:

The result of number of vowels is D3h.



2. Write an ALP to convert a hexadecimal number to binary pattern and store at 2000:3000h locations.

ALP:



Algorithm:

Step 1: Store the number in hexadecimal from into the num

Step 2: Then we use a number

Memory Address	Instruction	Hex code
1000 1001	Jmp HERE	EB,09
100B 100C 100D	HERE: LEA SI, num	BE 02 00
100E 100F	MOV CL,08H	B1 08
1010 1011	MOV AL,00H	B0 00
1012 1013	AGAIN: MOV BL,[SI]	8A 1C

1014 1015 1016	CMP BL,01H	80 FB 01
1017 1018	JNE NEXT	75 02

1019 101A	OR AL,01H	0C 01
101B 101C	NEXT: SHL AL,01H	D0 E0
101D	INC SI	46
101E 101F	LOOP AGAIN	E2 F2
1020 1021 1022	MOV res,AL	A2 0A 00
1023	HLT	F4

The Result:

The output is:

3.WAP to convert 8 bit BCD to Binary

Aim: Conversion from 8 bit BCD 10 Binary

ALP:

Algorithm:

Memory Address	Instruction	Hex code
1000 1001	Jmp HERE	EB,09
100B 100C 100D	HERE: LEA SI, num	BE 02 00
100E 100F	MOV CL,08H	B1 08
1010 1011	MOV AL,00H	B0 00
1012 1013	AGAIN: MOV BL,[SI]	8A 1C

1014 1015 1016	CMP BL,01H	80 FB 01
1017 1018	JNE NEXT	75 02
1019 101A	OR AL,01H	0C 01
101B 101C	NEXT: SHL AL,01H	D0 E0

101D	INC SI	46
101E 101F	LOOP AGAIN	E2 F2
1020 1021 1022	MOV res,AL	A2 0A 00
1023	HLT	F4

Result:

The Output is:

24 = 2 * 10 =20 + 4 =24

HINT: Use repeated addition for Multiplying , and for extracting the digit do MASKing with the help of AND Operation

SQUARE ROOT OF A NUMBER

CODE:

```
.MODEL SMALL

.STACK 100

.DATA

NUM1 DW 0010H

SQRT DW 01 DUP (?)


.CODE

START:


MOV AX,@DATA

MOV DS,AX

MOV AX,NUM1
```

```
XOR BX,BX
MOV BX,0001H
MOV CX,0001H
LOOP1:
   SUB AX,BX
   JZ LOOP2
   INC CX
   ADD BX,0002H
   JMP LOOP1
   INC CX

LOOP2:
   MOV SQRT,CX

INT 21H
END START
```

ALGORITHM

1. First the data is stord in the variable num1
2. The data is also loaded onto DS
3. BX and CX are initialized with 0001h
4. In loop1 bx is substracted from ax
5. if ax is 0 then jump to loop2
6. else increment cx

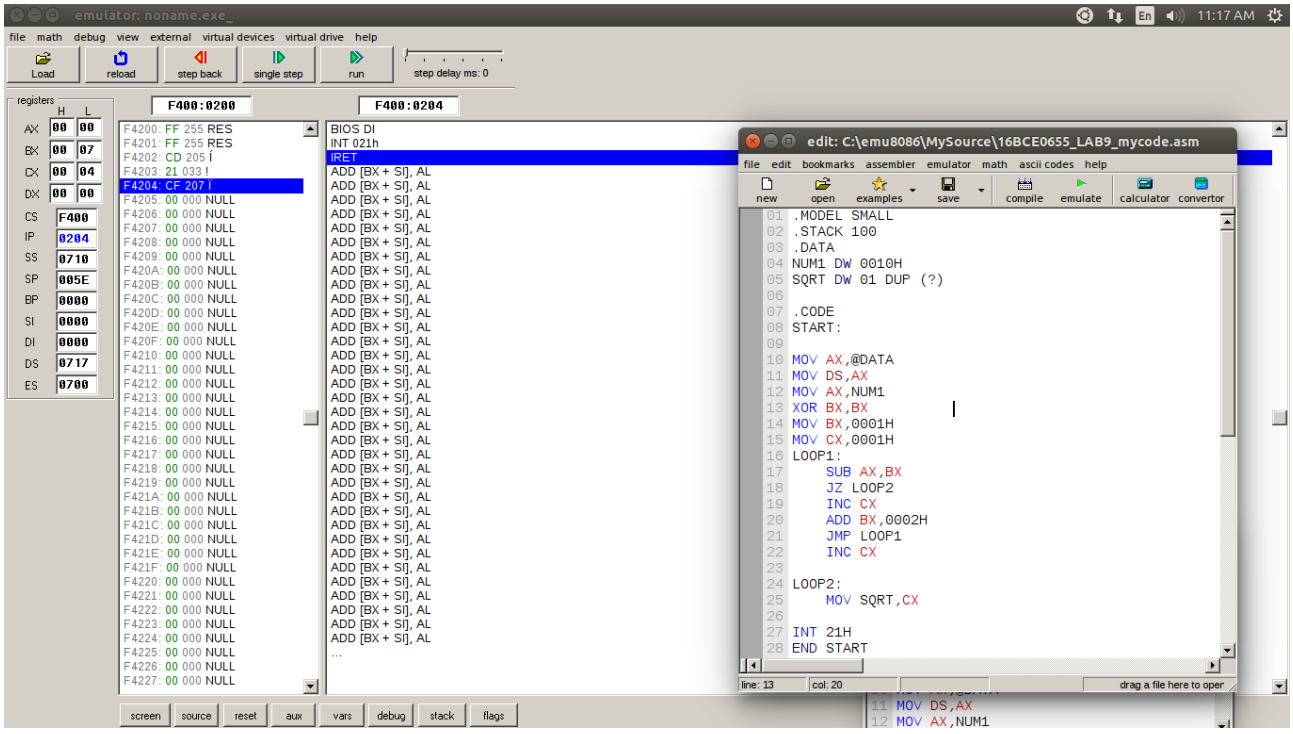
7. add 0002h to bx

8. jump to loop2

9. after loop1 completes increment cx by 1

10. In loop2 the value of cx is loaded onto cx

OUTPUT



INPUT: 10h == 16

OUTPUT: CX ==04H

RESULT: Hence the appropriate square root is found of the input

3. Fibonacci series

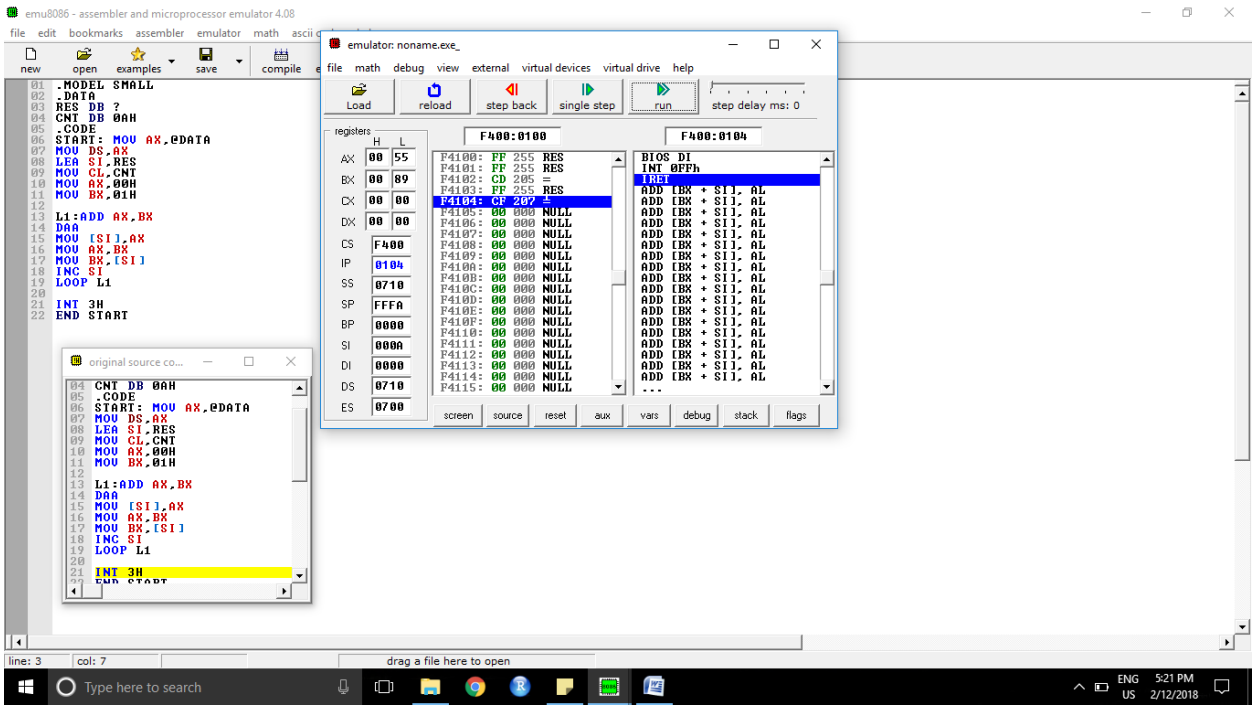
ALP:

```
.MODEL SMALL
.DATA
RES DB ?
CNT DB 0AH
.CODE
START: MOV AX,@DATA
MOV DS,AX
LEA SI,RES
MOV CL,CNT
MOV AX,00H
MOV BX,01H
L1:ADD AX,BX
DAA
MOV [SI],AX
MOV AX,BX
MOV BX,[SI]
INC SI
LOOP L1
INT 3H
END START
```

Algorithm:

1. We take to number of times the fibonaci series has to continue.
2. Then we initialize ax = 1 and bx=1 and res to store the result
3. After completing data segment we go for code segment
4. We run a loop where we add ax and bx

5. Put the value in source from ax
6. Put the value of ax in bx
7. Put the value of si into ax
8. Increase the value of si
9. Continue the loop
10. Thus terminate at the end and end the loop.



Input Sample:

10 is the number of times

Output Sample:

89 is the output.

Result:

The output of the Fibonacci series is 89h.

Masm

Opening of MASM

ftp://10.10.151.151/lab

copy and paste the MASM.ZIP file on your Desktop

Extract the file on the Desktop

Open DOS Box emulator:- it is the DOS virtual drive

Z:\>mount c ~ /desktop

Z:\>c:

C:\>cd MASM

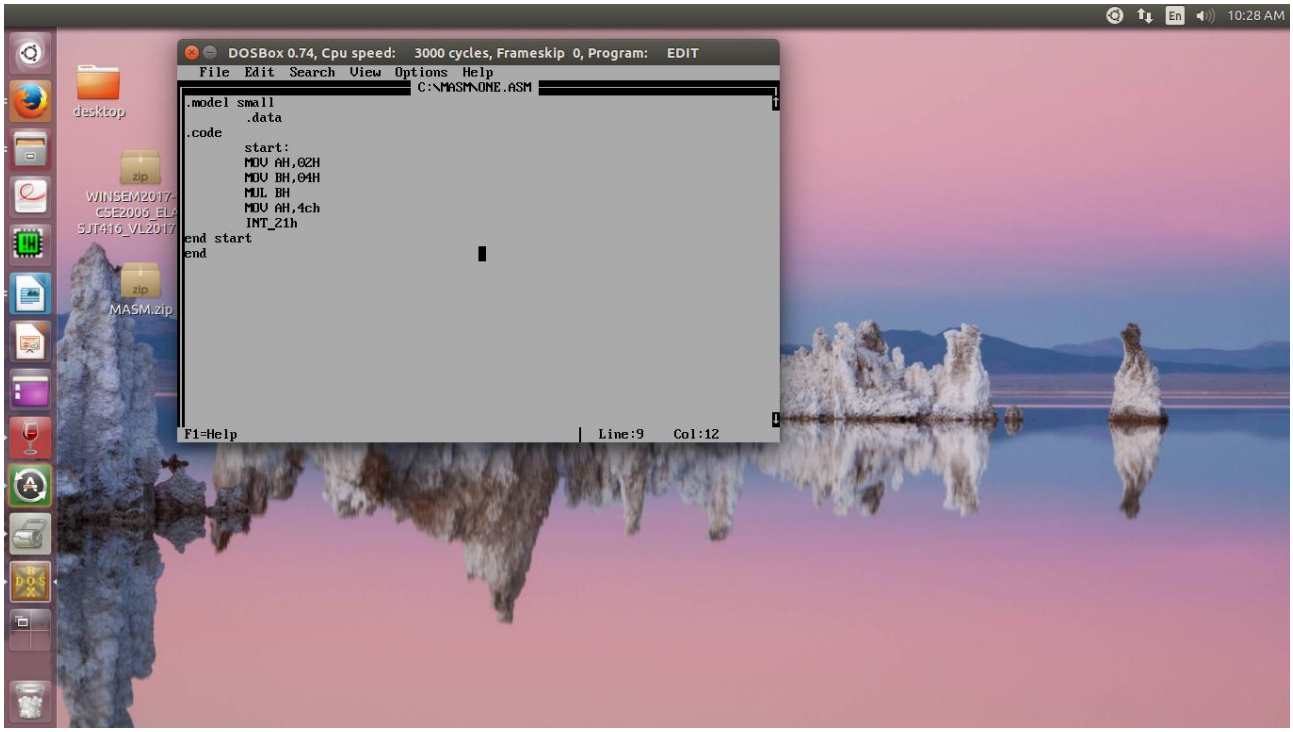
MASM:\>edit one.asm

Coding:

Multiplication of two numbers:-

```
.MODEL SMALL
.DATA
        ; No data defined
.CODE
        start:
                MOV AX,@DATA
                MOV DS, AX

                MOV AL, 09h
                MOV BL, 04H
                ADD AL,BL
```

```
        MOV AH, 4Ch ;terminate the process and returning to DOS or parent
                    process

        INT 21h

    END start

End
```



Then we save the program and run the code

Save the file.

MASM:\>Masm one.asm

Press ‘enter’ three times

MASM:\>Link one.obj

Press ‘enter’ three times

Debug one.exe

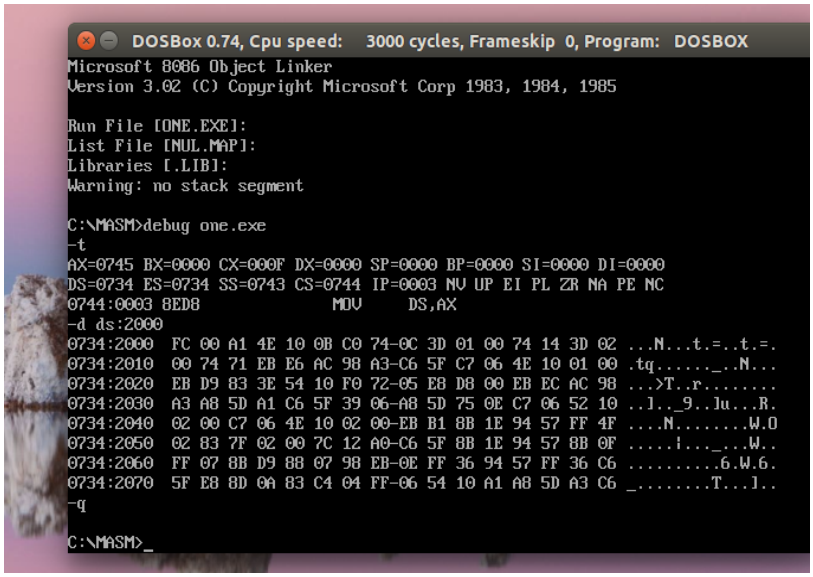
-t for line by line execution

-q for quit

-d ds:2000 to display memory content

-e ds:2000 to enter the data in memory

The output



KEYPAD:

PROCEDURE:

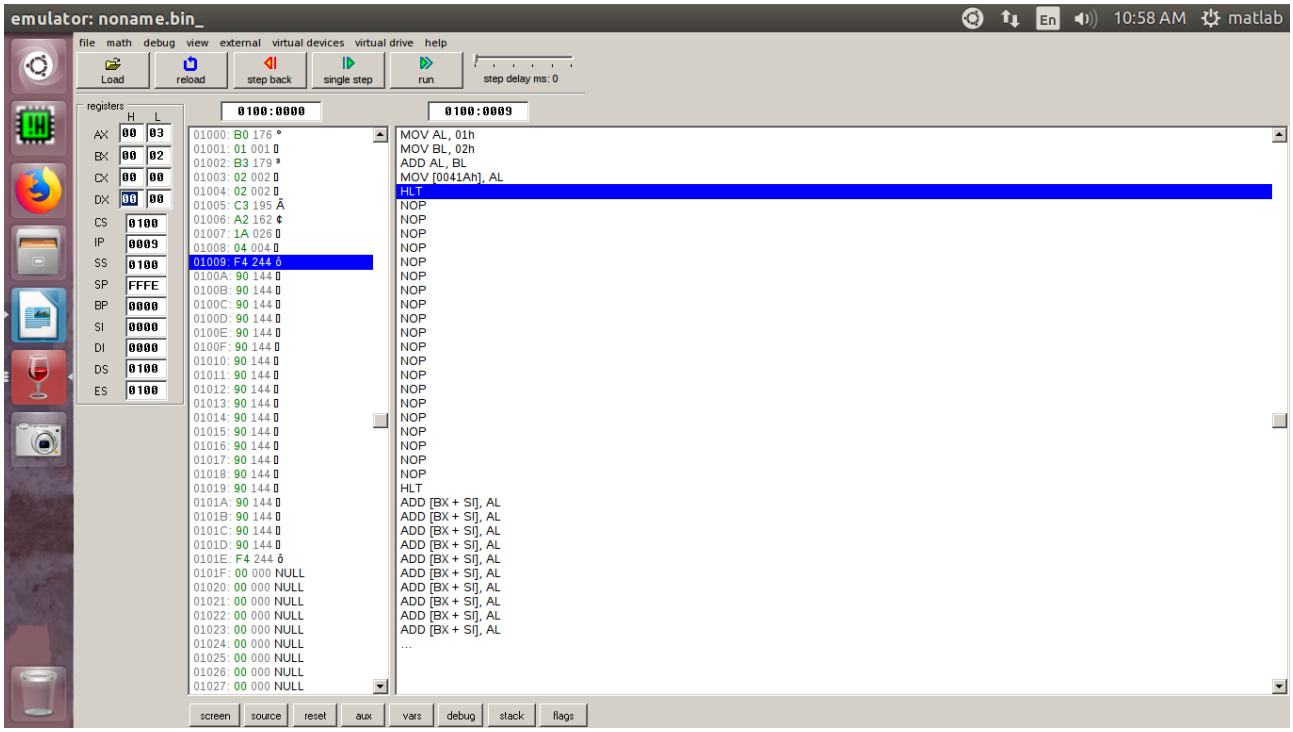
1. RST is pressed
2. After a – symbol is obtained SUB is pressed
3. The first address is loaded
4. NXT is pressed to go to the next instruction
5. The corresponding hex codes are loaded
6. Do this procedure till last hex code is F4
7. Press the RST button
8. Press GO button
9. Press EXC button
10. Again press the RST button
11. Press the SUB button
12. Enter the address of the result
13. Hence you get the result of the operation

1.

AIM: ADDITION OF TWO NUMBERS

ALP:

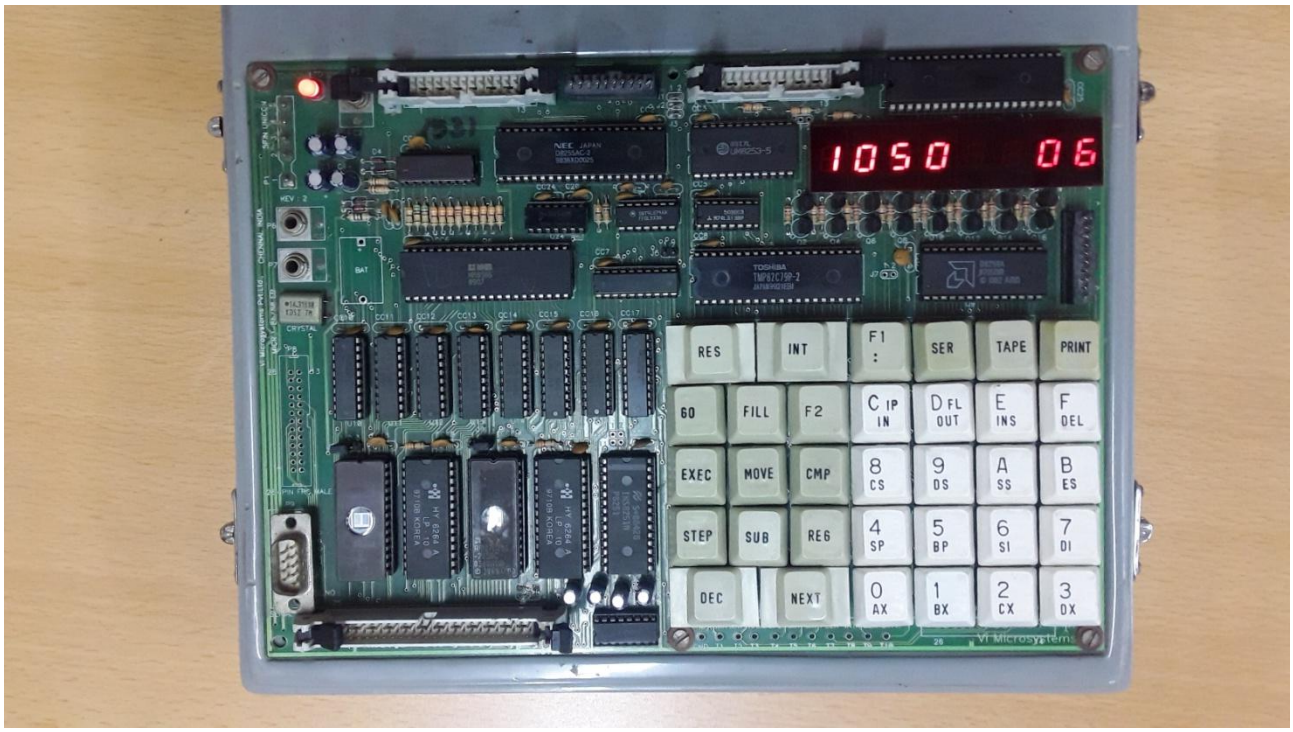
```
MOV AL, 01
MOV BL, 02
ADD AL,BL
MOV [1050],AL
HLT
```



ADDRESS	INSTRUCTION	HEX CODE
1000 1001	MOV AL,01H	B0 01
1002 1003	MOV BL,02H	B3 02
1004 1005	ADD AL,BL	02 C3
1006 1007 1008	MOV [1050],AL	A2 1A 04
1009	HLT	F4

THE OUTPUT:


1006 F4
RES
INT
F1 :
SER
TAPE
PRINT
GO
FILL
F2
C IP IN
D FL OUT
E INS
F DEL
EXEC
MOVE
CMP
8 CS
9 DS
A SS
B ES
STEP
SUB
REG
4 SP
5 BP
6 SI
7 DI
DEC
NEXT
0 AX
1 BX
2 CX
3 DX


1050 06
RES
INT
F1 :
SER
TAPE
PRINT
GO
FILL
F2
C IP IN
D FL OUT
E INS
F DEL
EXEC
MOVE
CMP
8 CS
9 DS
A SS
B ES
STEP
SUB
REG
4 SP
5 BP
6 SI
7 DI
DEC
NEXT
0 AX
1 BX
2 CX
3 DX

THE RESULT: 6H on addition.

2.

AIM: Find the factorial of a number

ALP:

MOV AL, 05

MOV BL, AL

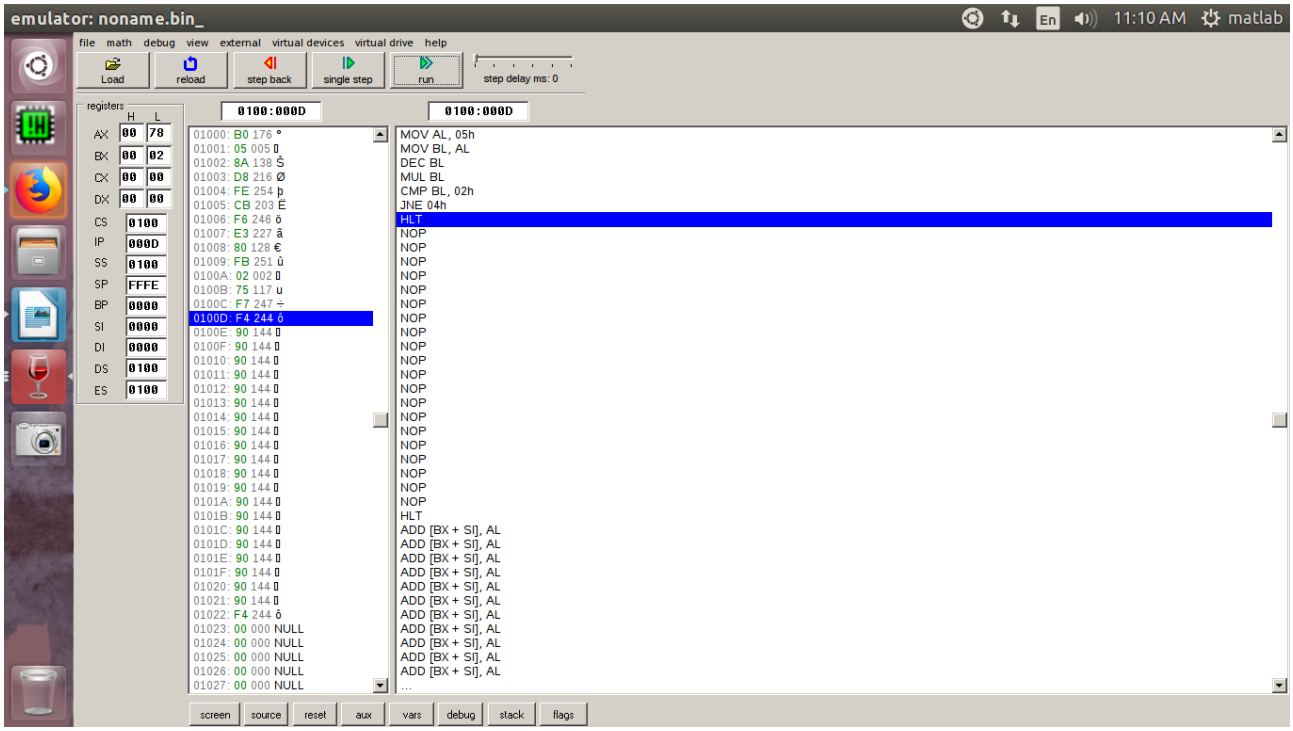
BACK: DEC BL

MUL BL

CMP BL,02H

JNE BACK

HLT



ADDRESS	INSTRUCTION	HEX CODE
1000 1001	MOV AL,05H	B0 05
1002 1003	MOV BL,AL	8A D8
1004 1005	DEC BL	FE CB
1006 1007	MUL BL	F6 E3
1008 1009 100A	CMP BL,02H	80 FB 02
100B 100C	JNE 02H	75 F7
100D	HLT	F4

THE OUTPUT:

THE RESULT: 78H is 5 FACTORIAL.

3.

AIM:SUBTRACTION OF TWO NUMBERS

ALP:

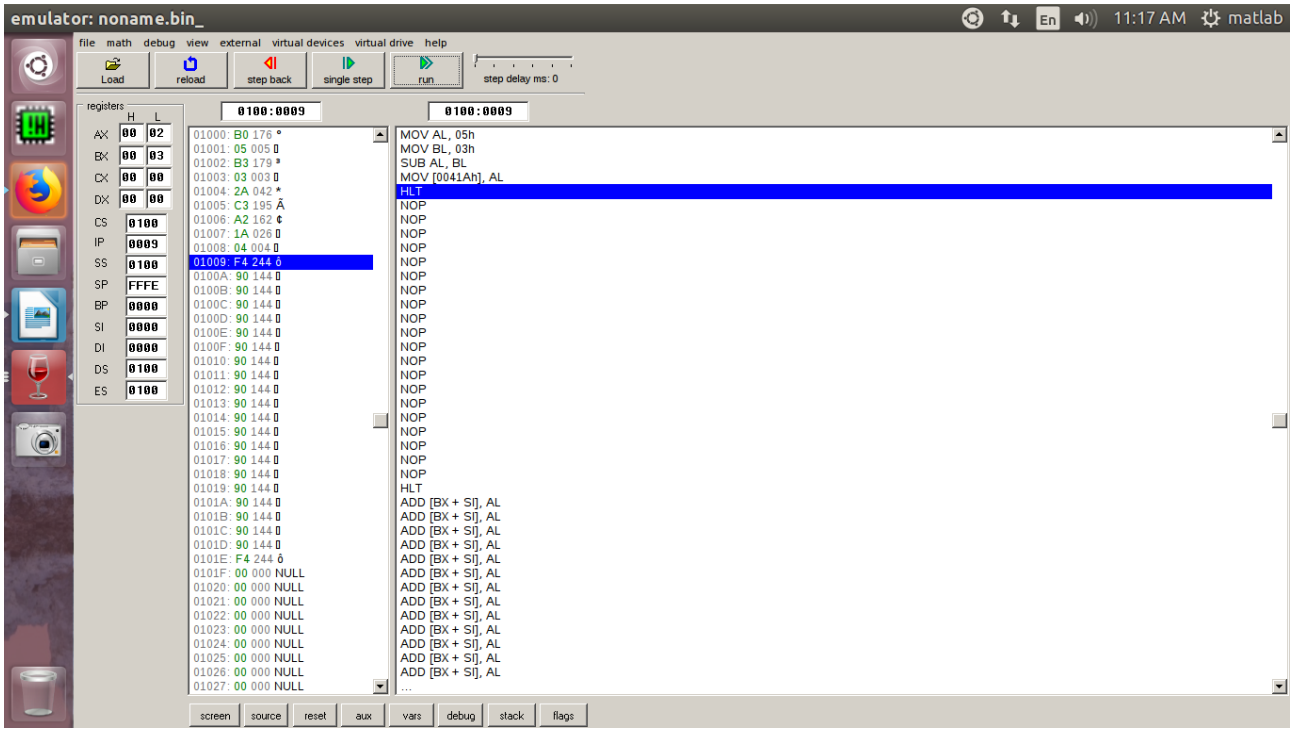
MOV AL, 05

MOV BL, 03

SUB AL,BL

MOV [1050],AL

HLT



ADDRESS	INSTRUCTION	HEX CODE
1000 1001	MOV AL,05H	B0 05
1002 1003	MOV BL,03H	B3 03
1004 1005	SUB AL,BL	2A C3
1006 1007 1008	MOV [1050],AL	A2 1A 04
1009	HLT	F4

THE OUTPUT:

THE RESULT: 2H on SUBTRACTION.