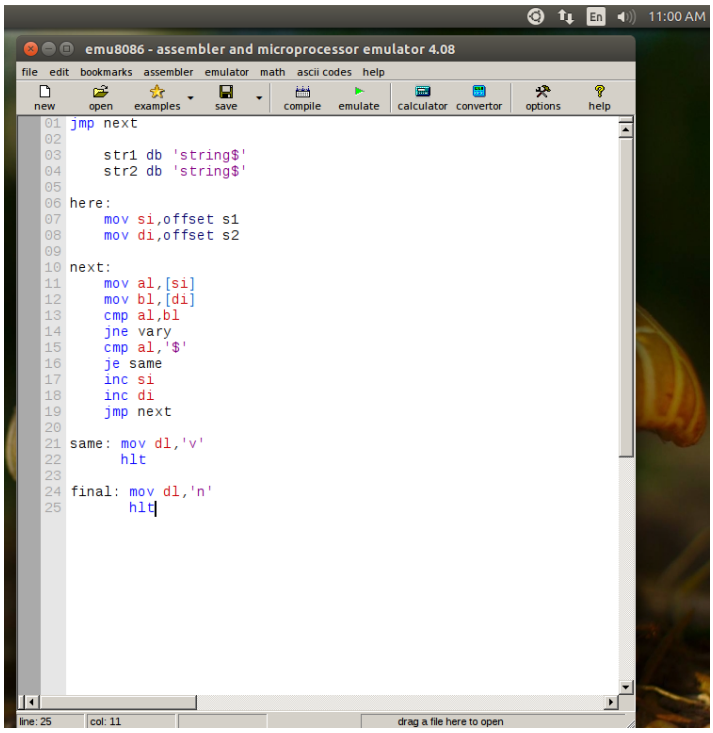
Microprocessor and Interfacing Lab
Experiment 3

Name: Om Ashish Mishra
Registration Number: 16BCE0789
Slot: B2

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

This question I have divided into 3 parts.

(i) 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

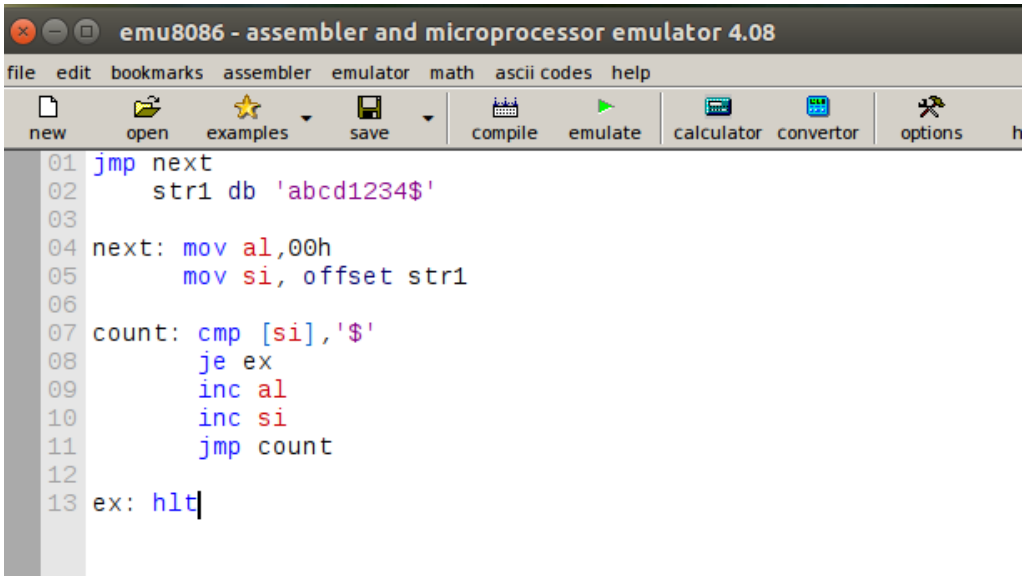
Sample Input: Ram,Ram

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

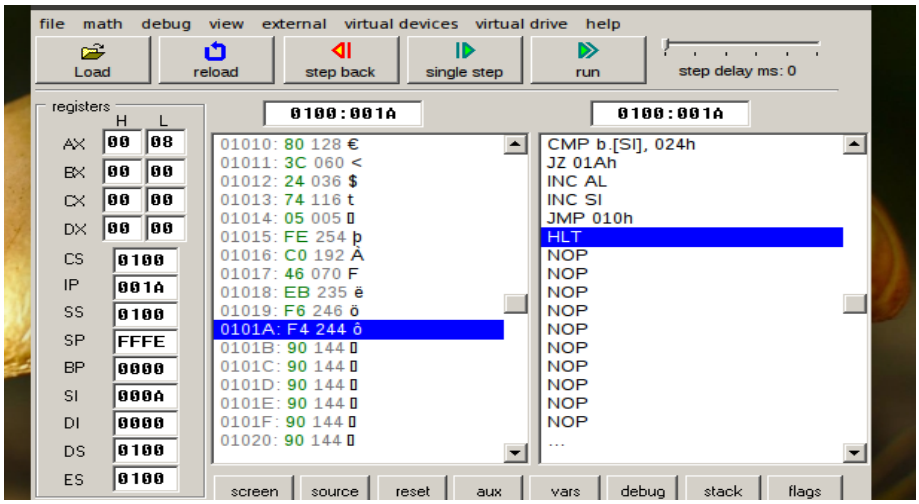
1006 1007	mov si,offset s1	F6,E3
1008 1009 100A	Count: cmp [si], '$'	80,FB,02
100B 100C	Je ex	75,F7
100D	Inc al	F4
1016 1017	Inc si	74 1A
1018 1019	Jmp count	3C 61
101A 101B	Ex: hlt	74 0B

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.

```
jmp next

    str db 'Education$'
    lea si,str
    mov bl,000h

next:
    mov al,[si]
    inc si
    cmp al,'$'
    je final
    cmp al,'a'
    je count
     cmp al,'e'
    je count
     cmp al,'i'
    je count
     cmp al,'o'
    je count
     cmp al,'u'
    je count
    jmp next

count: inc bl
       jmp next

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

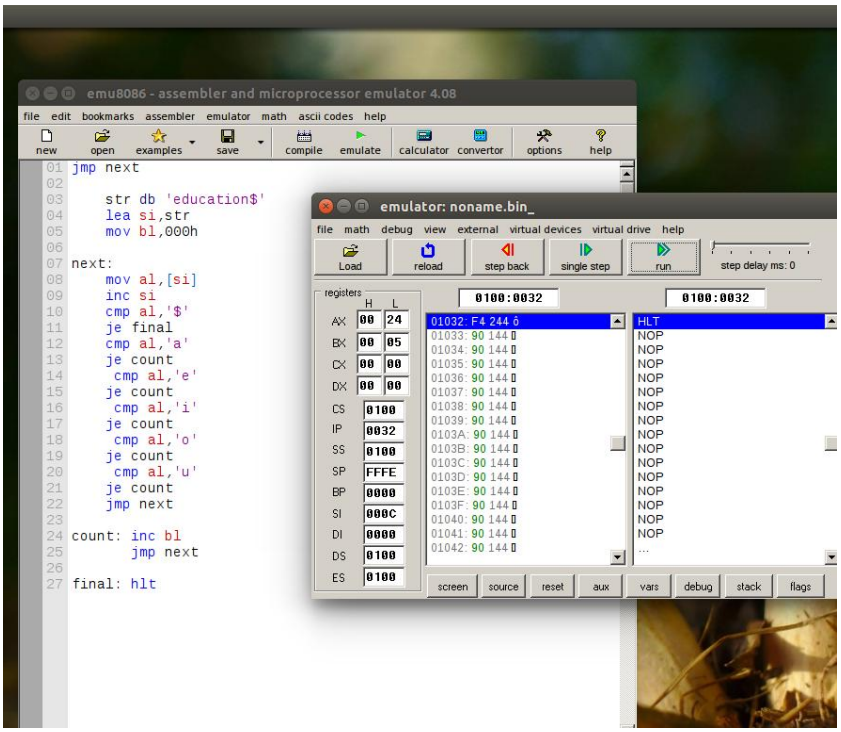
1004 1005	Lea si,str	FE,CB
1006 1007	Mov bl,000h	8A,4F
1011 1012	Next: mov al,[si]	8A,4f
1013	Inc si	46
1014 1015	Cmp al,'$'	3C,24
1016 1017	Je final	74 1A
1018 1019	Cmp al,'a'	3C 61
101A 101B	Je final	74 0B
101C 101D	Cmp al,'e'	3C 65
101E 101F	Je final	74 0E
1020 1021	Cmp al,'i'	3C 69
1022 1023	Je final	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

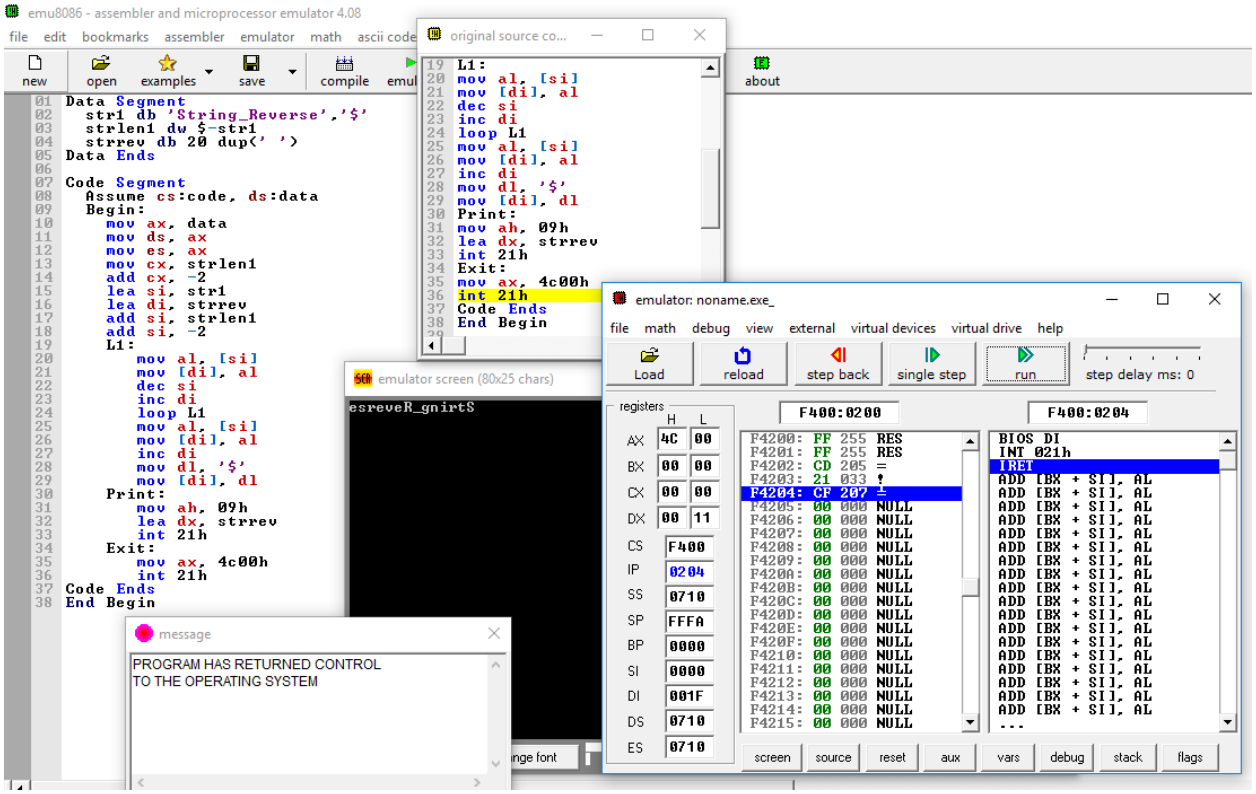
Aim: Reverse the given string

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

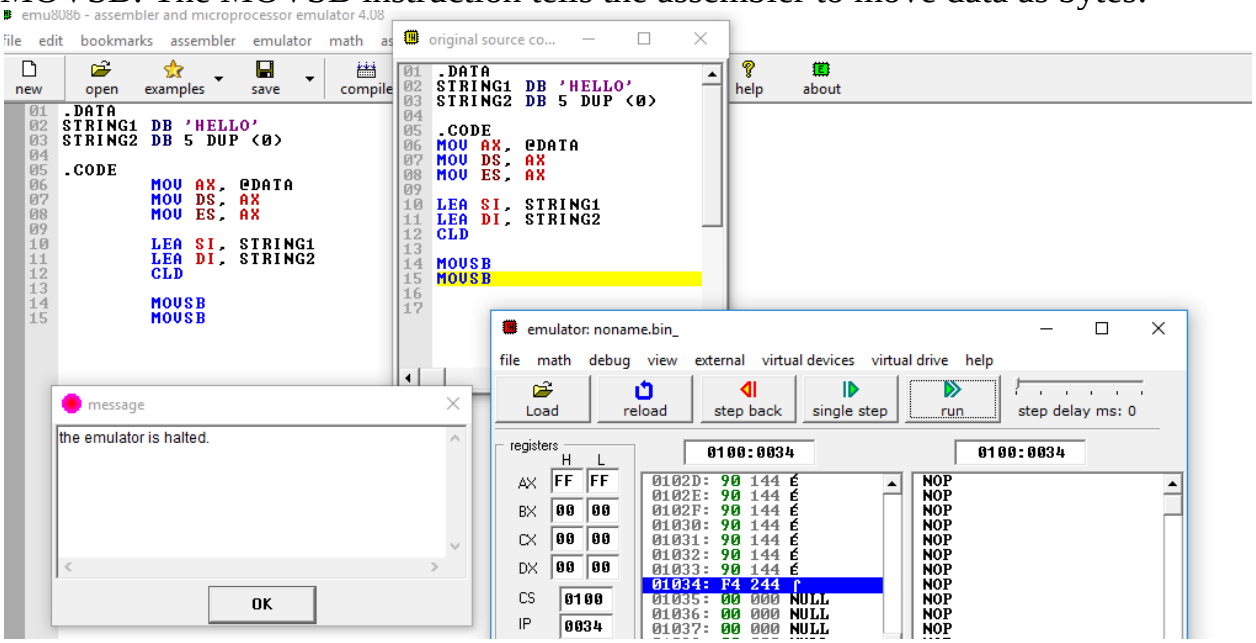
Sample Output: ereh era ew

Result:

The reverse string is given of the input.

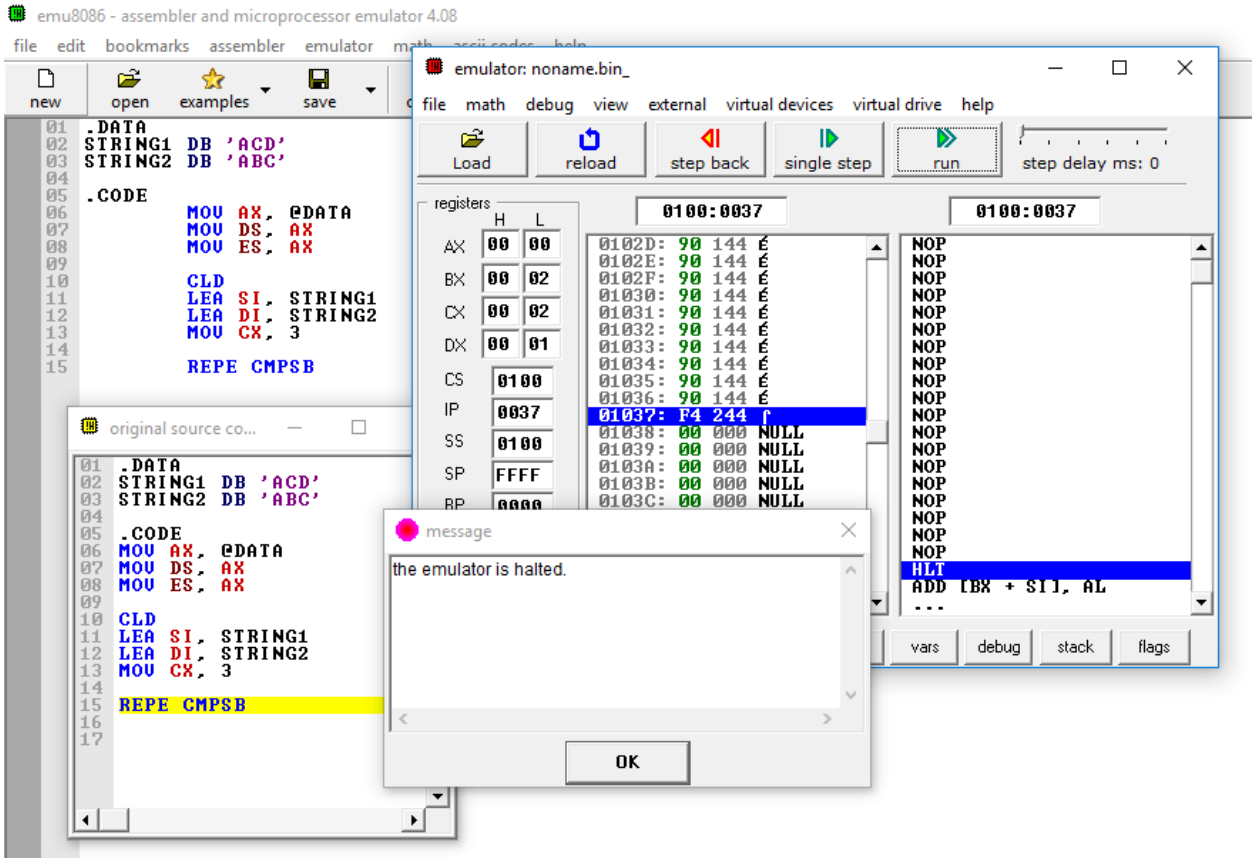
3. Analyse the string Instruction MOVSB, CMPS, SCAS etc.

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



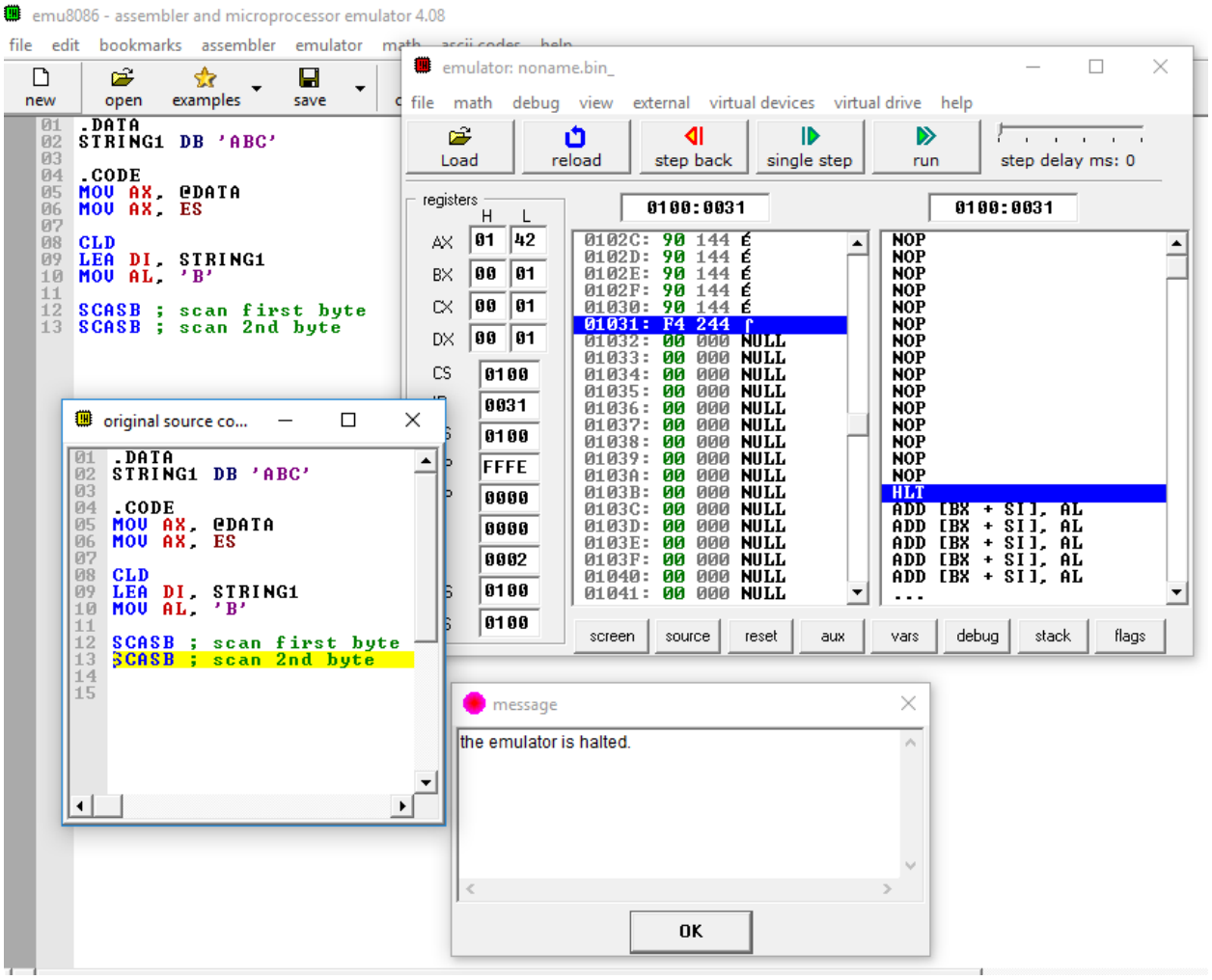
CMPS: This is used to compare the strings, words and double words

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.


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save
.DATA
STRING1 DB 'ABC'
.CODE
MOV AX, @DATA
MOV AX, ES
CLD
LEA DI, STRING1
MOV AL, 'B'
SCASB ; scan first byte
SCASB ; scan 2nd byte
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
H L
AX 01 42
BX 00 01
CX 00 01
DX 00 01
CS 0100
0031
0100
FFFE
0000
0000
0002
0100
0100
0100:0031
0100:0031
01031: F4 244
NOP
HLT
ADD [BX + SI], AL
screen source reset aux vars debug stack flags
original source co...
message
the emulator is halted.
OK