

<b>Name:</b>	Preeti Rambahal Prajapati
<b>Roll No:</b>	48
<b>Class/Sem:</b>	SE/IV
<b>Experiment No.:</b>	5
<b>Title:</b>	Program to display string in Lowercase.
<b>Date of Performance:</b>	
<b>Date of Submission:</b>	
<b>Marks:</b>	
<b>Sign of Faculty:</b>	

**Aim:** Program to display string in Lowercase.

**Theory:**

The program will take Uppercase string as input and convert it to lowercase string. Int 21h is a DOS interrupt. To use the DOS interrupt 21h load with the desired sub-function. Load other required parameters in other registers and make a call to INT 21h.

INT 21h/AH = 9

output of string at DS: • String must be terminated by "\$"

example :

org 100h

mov dx, offset msg

mov ah, 9

int 21h

ret

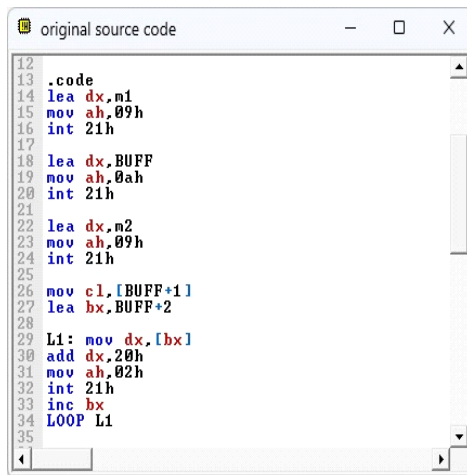
msg db "hello world \$"

INT 21h/AH = 0AH – input of string to DS:DX, first byte is buffer size, second byte is number of chars actually read this function does not add '\$' in the end of string to print using INT 21h/AH = 9 you must set dollar character at the end of it and start printing from address DS : DX + 2. The function does not allow to enter more characters than the specified buffer size.

### **Algorithm:**

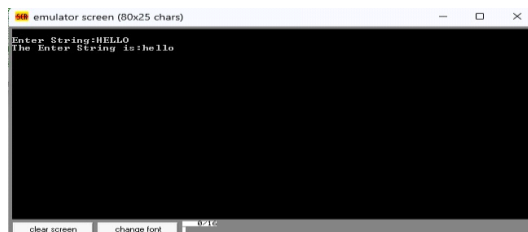
1. Start.
2. Initialize the Data Segment.
3. Display message -1.
4. Input the string.
5. Display message-2.
- 6 Take the character count in CX.
7. Point to the first character.
8. Convert it to Lowercase.
9. Display the character.
10. Decrement the character coun.
11. If not Zero, repeat from step 6.
12. To terminate the program, using the DOS interrupt:
  - 1) Initialize AH with 4CH
  - 2) Call interrupt INT 21H.
13. Stop.

### **Assembly Code:**



```
12
13 .code
14 lea dx, n1
15 mov ah, 09h
16 int 21h
17
18 lea dx, BUFF
19 mov ah, 0ah
20 int 21h
21
22 lea dx, n2
23 mov ah, 09h
24 int 21h
25
26 mov cl, [BUFF+1]
27 lea bx, BUFF+2
28
29 L1: mov dx, [bx]
30 add dx, 20h
31 mov ah, 02h
32 int 21h
33 inc bx
34 LOOP L1
35
```

## Output:



```
emulator screen (80x25 chars)
Enter String:HELLO
The Enter String is:hello
```

## Conclusion:

- Explain instruction AAA.

Ans. The AAA instruction, standing for “ASCII Adjust After Addition,” is a processor instruction used in x86 assembly language to adjust the result of a binary-coded decimal (BCD) addition operation. Here’s an explanation of the AAA instruction:

- Purpose: AAA is specifically designed to adjust the result in the AL register after adding two unpacked BCD digits together. It adjusts the AL register to represent a valid BCD result in the range of 0 to 9, and it also adjusts the flags accordingly.
- Usage: AAA typically follows an ADD instruction that adds two BCD digits together. After the addition operation, AAA examines the lower 4 bits (nibble) of the AL register. If this nibble contains a value between 0 and 9, indicating a valid BCD result, no adjustment is needed. However, if the nibble contains a value between 10 and 15, indicating an invalid BCD result, AAA adjusts the AL register, adds 6 to the result, and sets the carry flag (CF) and auxiliary carry flag (AF).
- Example: For example, if AL contains the value 0x13 after an addition operation (which is invalid in BCD), AAA will adjust AL to 0x19 and set the carry and auxiliary carry flags.
- Flags: AAA affects the carry flag (CF) and auxiliary carry flag (AF) to indicate the result’s validity and adjustment.

In summary, AAA is used to adjust the result of a BCD addition operation in the AL register, ensuring that it represents a valid BCD digit.

- Explain instruction AAS.

Ans. The AAS instruction, short for “ASCII Adjust After Subtraction,” is an x86 assembly language instruction used to adjust the result of a binary-coded decimal (BCD) subtraction operation. Here’s an explanation of the AAS instruction:

- Purpose: AAS is specifically designed to adjust the result in the AL register after subtracting one BCD digit from another. It ensures that the AL register represents a valid BCD result in the range of 0 to 9 and adjusts the flags accordingly.
- Usage: AAS is typically used after a subtraction operation involving two BCD digits. After the subtraction, AAS examines the lower 4 bits (nibble) of the AL register. If this nibble contains a value between 0 and 9, indicating a valid BCD result, no adjustment is needed. However, if the nibble contains a value between 10 and 15, indicating an invalid BCD result, AAS adjusts the AL register, subtracts 6 from the result, and sets the carry flag (CF) and auxiliary carry flag (AF).
- Example: For instance, if AL contains the value 0xFA after a subtraction operation (which is invalid in BCD), AAS will adjust AL to 0x00 and set the carry and auxiliary carry flags.
- Flags: AAS affects the carry flag (CF) and auxiliary carry flag (AF) to indicate the result’s validity and adjustment.

In summary, AAS is used to adjust the result of a BCD subtraction operation in the AL register, ensuring that it represents a valid BCD digit.