**JAWAHARLAL NEHRU UNIVERSITY**



ACADEMIC YEAR: 2021-22

SCHOOL OF COMPUTER & SYSTEM SCIENCES

Project: STRING OPERATION

Subject: Computer Architecture (CS-405)

Submitted To:Ashutosh Shrivastava sir

Submitted By : Naziya Khalid(21/10/JC/036)

**General Purpose Register in 8086**

GROUP MEMBERS : Harshita Chourasiya(21/10/JC/035)

Naziya Khalid(21/10/JC/036)

Himani Saini(21/10/JC/001)

**AX –** This is the accumulator. It is of 16 bits and is divided into two 8-bit registers AH and AL to also perform 8-bit instructions.  
It is generally used for arithmetical and logical instructions but in 8086 microprocessor it is not mandatory to have accumulator as the destination operand.

Example:

ADD AX, AX (AX = AX + AX)

**BX –** This is the base register. It is of 16 bits and is divided into two 8-bit registers BH and BL to also perform 8-bit instructions.  
It is used to store the value of the offset.

Example:

MOV BL, [500] (BL = 500H)

**CX –** This is the counter register. It is of 16 bits and is divided into 8-bit registers CH and CL to also perform 8-bit instructions.  
It is used in looping and rotation.

Example:

MOV CX, 0005

LOOP

**DX –** This is the data register. It is of 16 bits and is divided into two 8-bit registers DH and DL to also perform 8-bit instructions.  
It is used in multiplication an input/output port addressing.

Example:

MUL BX (DX, AX = AX \* BX)

**SP –** This is the stack pointer. It is of 16 bits.

It points to the topmost item of the stack. If the stack is empty the stack pointer will be (FFFE)H. It’s offset address relative to stack segment.

**BP –** This is the base pointer. It is of 16 bits.  
It is primary used in accessing parameters passed by the stack. It’s offset address relative to stack segment.

**SI –** This is the source index register. It is of 16 bits.  
 It is used in the pointer addressing of data and as a source in some string related operations. It’s offset is relative to data segment.

**DI –** This is the destination index register. It is of 16 bits.  
It is used in the pointer addressing of data and as a destination in some string related operations.It’s offset is relative to extra segment.

STRING OPERATION

* In this program , We take the value from the user in form of string . Then we calculate that how many Characters are given in the String.
* Accordingly we give the output as Even and Odd. If the given strings has even no. of characters it will give Even as output and vice versa.
* And it also perform the searching operation where we ask the user to find any character from their given string and we get the place where our character is placed.

Instructions used in the Program

* **MOV** − Used to copy the byte or word from the provided source to the provided destination.
* **DEC** − Used to decrement the provided byte/word by 1.
* **DIV** − Used to divide the unsigned word by byte or unsigned double word by word.
* **JE/JZ** − Used to jump if equal/zero flag ZF = 1
* **INT** − Used to interrupt the program during execution and calling service specified.
* **RET** − Used to return from the procedure to the main program.
* **JMP** − Used to jump to the provided address to proceed to the next instruction.
* **CMP** − Used to compare 2 provided byte/word.
* **CALL** − Used to call a procedure and save their return address to the stack.
* **ADD** − Used to add the provided byte to byte/word to word.
* **INC** − Used to increment the provided byte/word by 1.

**CODE**

dosseg

.model small

.stack 100h

.data

string db 10 dup('$')

msg1 db 'Enter String: $'

msg2 db 'Maximum of length is Reached $'

msg3 db 'Enter a character to find $'

msgFound db 'Character is found at $'

msgNotFound db 'Character is not found $'

msgEven db 'Even $'

msgOdd db 'Odd $'

.code

main proc

mov ax, @data

mov ds, ax

mov bl, 1    ; counts the length of string

mov dx, offset msg1

mov ah, 9

int 21h

mov si, offset string

l1:

mov ah, 1

int 21h

cmp al, 13

je EnterKeyPressed

cmp bl, 10

je LengthReached

mov [si], al

inc si

inc bl

jmp l1

LengthReached:

call enterbutton

mov ah, 9

mov dx, offset msg2

int 21h

call enterbutton

mov dx, offset msgEven

mov ah, 9

int 21h

jmp AskCharacter

EnterKeyPressed:

call evenodd

jmp AskCharacter

AskCharacter:

call enterbutton

mov dx, offset msg3

mov ah, 9

int 21h

mov ah, 1

int 21h

mov bl, 1

mov si, offset string

l2:

cmp [si],al

je PrintFound

cmp bl, 10

je NotFound

inc bl

inc si

jmp l2

PrintFound:

call enterbutton

mov dx, offset msgFound

mov ah, 9

int 21h

mov dx, 0

mov dl, bl

add dl, 48

mov ah, 2

int 21h

call enterbutton

mov ah, 4ch

int 21h

NotFound:

call enterbutton

mov dx, offset msgNotFound

mov ah, 9

int 21h

mov ah, 4ch

int 21h

main endp

enterbutton proc

mov ah, 2

mov dl, 13

int 21h

mov ah, 2

mov dl, 10

int 21h

ret

enterbutton endp

evenodd proc

dec bl

mov ah, 0

mov al, bl

mov bl, 2

div bl

cmp ah,0

je IsEven

mov dx, offset msgOdd

mov ah, 9

int 21h

ret

IsEven:

mov dx, offset msgEven

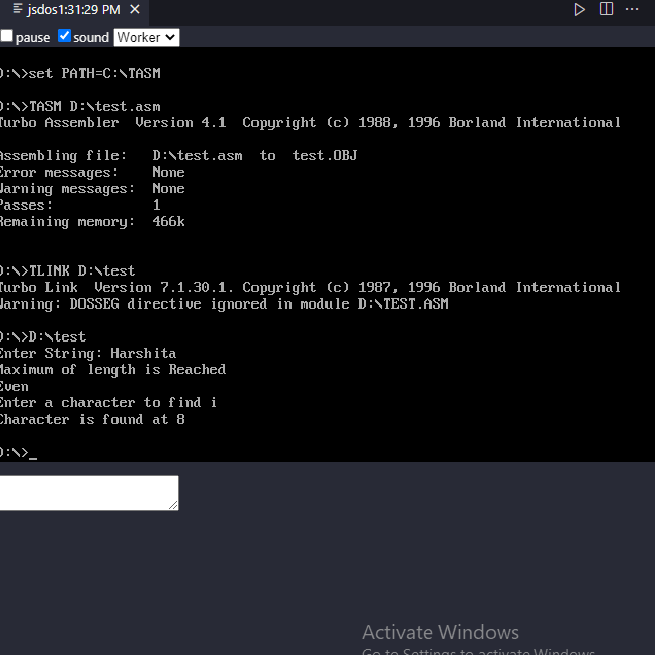
mov ah, 9

int 21h

ret

evenodd endp

end main



OUTPUT