# TASK 1:

**If even number is pressed it displays Even else Odd (0-9)**

org 0x0100

jmp start

; Keyboard interrupt service routine

kbisr:

push ax

push es

mov ax, 0xb800

mov es, ax ; point es to video memory

in al, 0x60 ; read a char from the keyboard port

; Check if the key is a even

cmp al, 0x03

je even

cmp al, 0x05

je even

cmp al, 0x07

je even

cmp al, 0x09

je even

cmp al, 0x0B

je even

jmp odd

even:

mov byte [es:3840], 'E'

mov byte [es:3842], 'V'

mov byte [es:3844], 'E'

mov byte [es:3846], 'N'

jmp done ; Leave interrupt routine

odd:

mov byte [es:3840], 'O'

mov byte [es:3842], 'D'

mov byte [es:3844], 'D'

mov byte [es:3846], ' '

jmp done

done:

mov al, 0x20

out 0x20, al ; send EOI to PIC

pop es

pop ax

iret

start:

xor ax, ax

mov es, ax ; point es to IVT base

cli ; disable interrupts

mov word [es:9\*4], kbisr ; store offset at n4

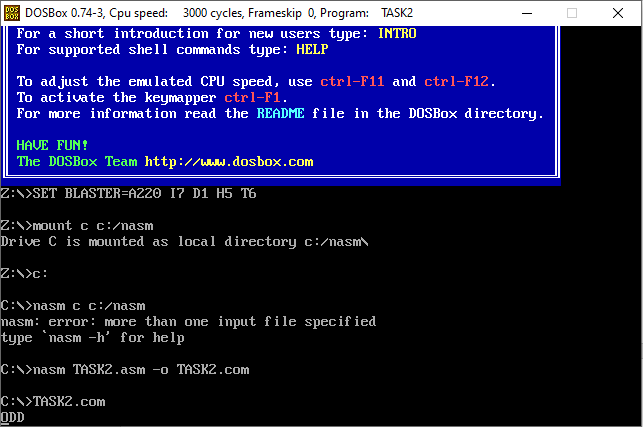
mov [es:9\*4+2], cs ; store segment at n4+2

sti ; enable interrupts

l1:

jmp l1 ; infinite loop

# OUTPUT:



# TASK 2:

**Notepad reading all alphabets with space, backspace and enter (next line) functionality**

org 0x0100

jmp start

currentPosition dw 0 ; Store current position in video memory

currentLine dw 0 ; Store current line number

; Keyboard interrupt service routine

kbisr:

push ax

push es

mov ax, 0xb800

mov es, ax ; point es to video memory

in al, 0x60 ; read a char from the keyboard port

; Check for specific keys

cmp al, 0x10 ; 'Q' key

je near printQ

cmp al, 0x11 ; 'W' key

je near printW

cmp al, 0x12 ; 'E' key

je near printE

cmp al, 0x13 ; 'R' key

je near printR

cmp al, 0x14 ; 'T' key

je near printT

cmp al, 0x15 ; 'Y' key

je near printY

cmp al, 0x16 ; 'U' key

je near printU

cmp al, 0x17 ; 'I' key

je near printI

cmp al, 0x18 ; 'O' key

je near printO

cmp al, 0x19 ; 'P' key

je near printP

cmp al, 0x1E ; 'A' key

je near printA

cmp al, 0x1F ; 'S' key

je near printS

cmp al, 0x20 ; 'D' key

je near printD

cmp al, 0x21 ; 'F' key

je near printF

cmp al, 0x22 ; 'G' key

je near printG

cmp al, 0x23 ; 'H' key

je near printH

cmp al, 0x24 ; 'J' key

je near printJ

cmp al, 0x25 ; 'K' key

je near printK

cmp al, 0x26 ; 'L' key

je near printL

cmp al, 0x2C ; 'Z' key

je near printZ

cmp al, 0x2D ; 'X' key

je near printX

cmp al, 0x2E ; 'C' key

je near printC

cmp al, 0x2F ; 'V' key

je near printV

cmp al, 0x30 ; 'B' key

je near printB

cmp al, 0x31 ; 'N' key

je near printN

cmp al, 0x32 ; 'M' key

je near printM

cmp al, 0x39 ; Space key

je near printSpace

cmp al, 0x0E ; Backspace key

je near backspace

cmp al, 0x1C ; Enter key

je near enterKey

jmp done

printQ:

mov di, [currentPosition]

mov byte [es:di], 'Q' ; Display 'Q'

jmp moveNext

printW:

; Display 'W' at the current position

mov di, [currentPosition]

mov byte [es:di], 'W' ; Display 'W'

jmp moveNext

printE:

; Display 'E' at the current position

mov di, [currentPosition]

mov byte [es:di], 'E' ; Display 'E'

jmp moveNext

printR:

; Display 'R' at the current position

mov di, [currentPosition]

mov byte [es:di], 'R' ; Display 'R'

jmp moveNext

printT:

; Display 'T' at the current position

mov di, [currentPosition]

mov byte [es:di], 'T' ; Display 'T'

jmp moveNext

printY:

; Display 'Y' at the current position

mov di, [currentPosition]

mov byte [es:di], 'Y' ; Display 'Y'

jmp moveNext

printU:

; Display 'U' at the current position

mov di, [currentPosition]

mov byte [es:di], 'U' ; Display 'U'

jmp moveNext ;

printI:

; Display 'I' at the current position

mov di, [currentPosition]

mov byte [es:di], 'I' ; Display 'I'

jmp moveNext

printO:

; Display 'O' at the current position

mov di, [currentPosition]

mov byte [es:di], 'O' ; Display 'O'

jmp moveNext

printP:

; Display 'P' at the current position

mov di, [currentPosition]

mov byte [es:di], 'P' ; Display 'P'

jmp moveNext

printA:

; Display 'A' at the current position

mov di, [currentPosition]

mov byte [es:di], 'A' ; Display 'A'

jmp moveNext

printS:

; Display 'S' at the current position

mov di, [currentPosition]

mov byte [es:di], 'S' ; Display 'S'

jmp moveNext

printD:

; Display 'D' at the current position

mov di, [currentPosition]

mov byte [es:di], 'D' ; Display 'D'

jmp moveNext

printF:

; Display 'F' at the current position

mov di, [currentPosition]

mov byte [es:di], 'F' ; Display 'F'

jmp moveNext

printG:

; Display 'G' at the current position

mov di, [currentPosition] ;

mov byte [es:di], 'G' ; Display 'G'

jmp moveNext

printH:

; Display 'H' at the current position

mov di, [currentPosition] ;

mov byte [es:di], 'H' ; Display 'H'

jmp moveNext

printJ:

; Display 'J' at the current position

mov di, [currentPosition] ;

mov byte [es:di], 'J' ; Display 'J'

jmp moveNext

printK:

; Display 'K' at the current position

mov di, [currentPosition]

mov byte [es:di], 'K' ; Display 'K'

jmp moveNext

printL:

; Display 'L' at the current position

mov di, [currentPosition]

mov byte [es:di], 'L' ; Display 'L'

jmp moveNext

printZ:

; Display 'Z' at the current position

mov di, [currentPosition]

mov byte [es:di], 'Z' ; Display 'Z'

jmp moveNext

printX:

; Display 'X' at the current position

mov di, [currentPosition]

mov byte [es:di], 'X' ; Display 'X'

jmp moveNext

printC:

; Display 'C' at the current position

mov di, [currentPosition]

mov byte [es:di], 'C' ; Display 'C'

jmp moveNext

printV:

; Display 'V' at the current position

mov di, [currentPosition]

mov byte [es:di], 'V' ; Display 'V'

jmp moveNext

printB:

; Display 'B' at the current position

mov di, [currentPosition]

mov byte [es:di], 'B' ; Display 'B'

jmp moveNext

printN:

; Display 'N' at the current position

mov di, [currentPosition]

mov byte [es:di], 'N' ; Display 'N'

jmp moveNext

printM:

; Display 'M' at the current position

mov di, [currentPosition]

mov byte [es:di], 'M' ; Display 'M'

jmp moveNext

backspace:

; Clear the character at the current position

sub byte [currentPosition], 2

mov di, [currentPosition] ;

mov byte [es:di], ' ' ; Display space character

printSpace:

; Display space at the current position

mov di, [currentPosition]

mov byte [es:di], ' ' ; Display space character

jmp moveNext

enterKey:

; Move to the next line

add word [currentLine], 1 ; Increment the current line number

mov word [currentPosition], [currentLine] ; Set the cursor to the beginning of the next line

shl word [currentPosition], 1 ; Multiply the current line number by 2 (each character is 2 bytes)

moveNext:

add byte [currentPosition], 2 ; Move to the next character position

done:

mov al, 0x20

out 0x20, al ; send EOI to PIC

pop es

pop ax

iret

start:

xor ax, ax

mov es, ax ; point es to IVT base

cli ; disable interrupts

mov word [es:9\*4], kbisr ; store offset at n4

mov [es:9\*4+2], cs ; store segment at n4+2

sti ; enable interrupts

l1:

jmp near l1 ; infinite loop

# OUTPUT:

