

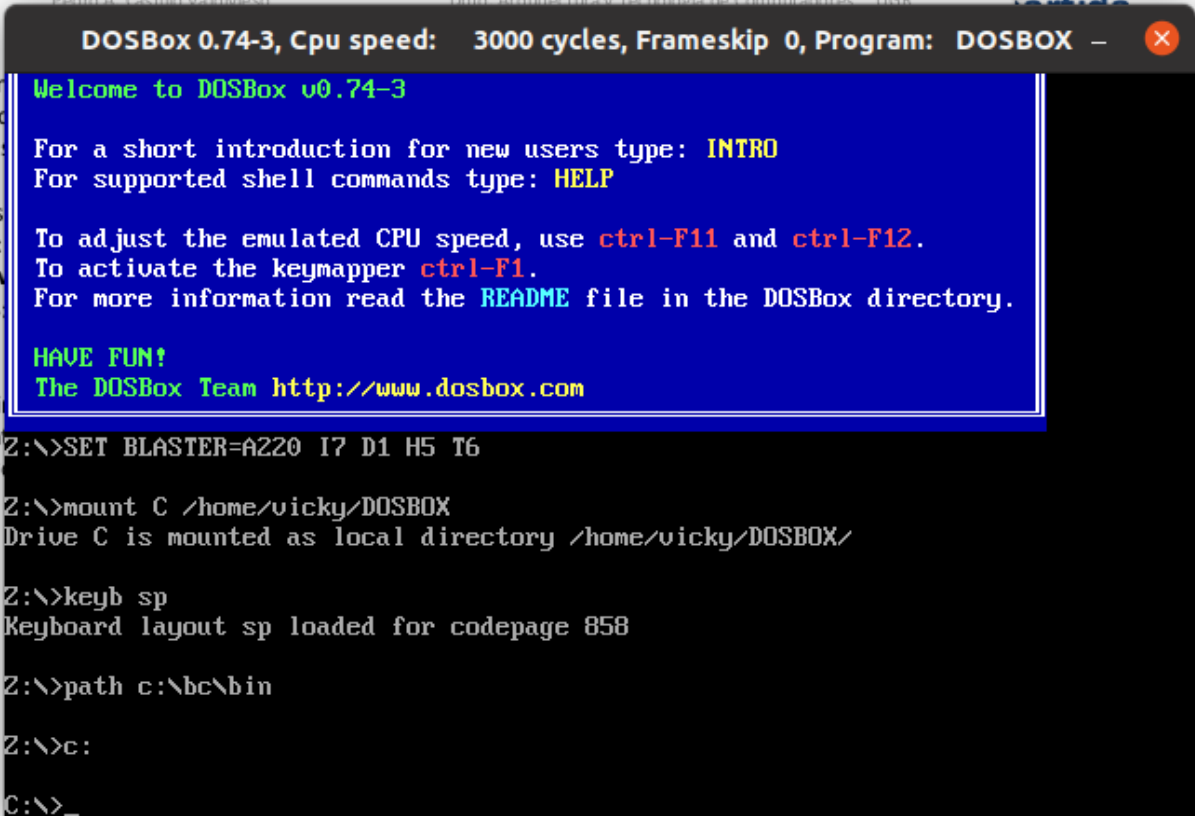
# Seminario 1. Programación de dispositivos a bajo nivel

## Instalación y configuración de DOSBOX

Una vez instalado DOSBOX, descargado a través de la página web oficial <http://www.dosbox.com/>, procedemos a modificar el archivo de configuración de DOSBOX. Se trata de montar automáticamente la unidad c en la carpeta DOSBOX que indicamos y, además, también se lo modificamos para poner el teclado en español. Para ello, añadimos las siguientes cuatro líneas al final del archivo:

```
mount C /home/vicky/DOSBOX
keyb sp
path c:\bc\bin
c:
```

Si este proceso se ha realizado con éxito, al iniciar DOSBOX nos tendría que aparecer automáticamente la siguiente pantalla:



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX -
Welcome to DOSBox v0.74-3
For a short introduction for new users type: INTRO
For supported shell commands type: HELP
To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.
HAVE FUN!
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>mount C /home/vicky/DOSBOX
Drive C is mounted as local directory /home/vicky/DOSBOX/
Z:\>keyb sp
Keyboard layout sp loaded for codepage 858
Z:\>path c:\bc\bin
Z:\>c:
C:\>_
```

# Ejecución de diferentes aplicaciones de ejemplo

## Juegos clásicos

Como bien nos explicaron en clase, dentro de DOSBOX podemos reproducir y jugar a juego clásicos. Para ello, simplemente necesitamos entrar en la carpeta donde se encuentra el archivo .EXE del juego y ejecutarlo. A continuación, podemos ver algunos ejemplos de estos juegos.

### VBALL

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX -
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.
HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount C /home/vicky/DOSBOX
Drive C is mounted as local directory /home/vicky/DOSBOX/

Z:\>keyb sp
Keyboard layout sp loaded for codepage 858

Z:\>path c:\bc\bin

Z:\>c:

C:\>cd JUEGOS

C:\JUEGOS>cd JUEGOS~1

C:\JUEGOS\JUEGOS~1>cd VBALL

C:\JUEGOS\JUEGOS~1\VBALL>VBALL.EXE
```



## GOLDEN

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AXE
C:\JUEGOS\JUEGOS~1>dir
Directory of C:\JUEGOS\JUEGOS~1\
.                <DIR>                11-03-2022  11:59
..               <DIR>                11-03-2022  11:59
ARKANO~1         <DIR>                26-02-2020  11:01
GOLDEN           <DIR>                05-03-2022   9:04
LIVIN           <DIR>                05-03-2022   9:09
VBALL           <DIR>                06-11-2014   9:24
0 File(s)        0 Bytes.
6 Dir(s)         262,111,744 Bytes free.

C:\JUEGOS\JUEGOS~1>cd GOLDEN

C:\JUEGOS\JUEGOS~1\GOLDEN>GOLD.EXE
GOLDEN AXE

1 - EGA (16 Colors)
2 - CGA, MCGA, or Tandy (4 colors)
3 - Hercules Graphics Adapter
4 - VGA (256 colors)
5 - Tandy (16 colors)
ESC - Back to DOS

```

Seleccionamos la opción 2 al azar.



Aunque pidió una contraseña para poder jugar y no la teníamos, el programa se ejecutó sin problemas.

## “Hola mundo”

En primer lugar, tenemos un archivo proporcionado por el profesor en el que se muestra la palabra *hola* y está programado en ensamblador. Partiremos de ese código para modificarlo y que muestre el mensaje *Hola mundo!* siete veces.

Código que muestra una sola vez el mensaje (mostrado mediante \$ EDITAR.BAT HOLA):

```

[.]
pila segment stack 'stack'
dw 100h dup (?)
pila ends
datos segment 'data'
msg db 'Hola mundo!$'
datos ends
codigo segment 'code'
assume cs:codigo, ds:datos, ss:pila
main PROC
    mov ax,datos
    mov ds,ax

    mov dx,OFFSET msg
    mov ah,9
    int 21h

    mov ax,4C00h
    int 21h
main ENDP
codigo ends
END main

1:1
int86(0x10, &inress, &outress);
}

int main(){
    int tmp;

    printf("\nPulsa una tecla... ");
    tmp = getch();
}

10:1

```

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Compilamos el programa y lo ejecutamos con:

\$ C.BAT HOLA

\$ HOLA.EXE

```

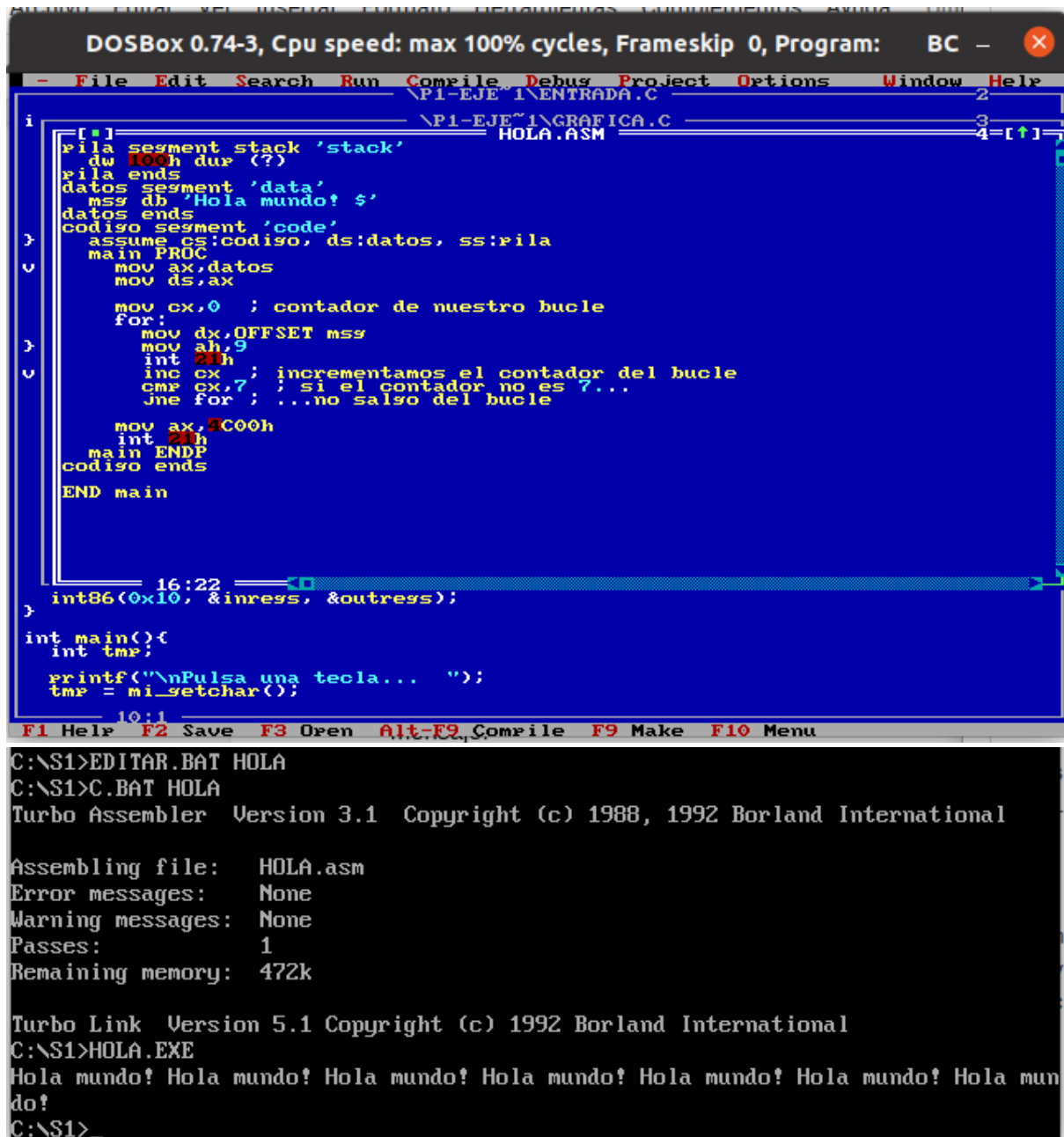
C:\S1>C.BAT HOLA
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International

Assembling file:   HOLA.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  472k

Turbo Link Version 5.1 Copyright (c) 1992 Borland International
C:\S1>HOLA.EXE
Hola mundo!
C:\S1>

```

A continuación, modificamos nuevamente el archivo para que se muestre 7 veces el mensaje. Volvemos a compilar y a ejecutar. Observamos que efectivamente aparece el mensaje repetido 7 veces.



DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Program: BC

File Edit Search Run Compile Debug Project Options Window Help

\P1-EJE\1\ENTRADA.C 2  
 \P1-EJE\1\GRAFICA.C 3  
 HOLA.ASM 4

```

[.]
rila segment stack 'stack'
dw 100h dup (?)
rila ends
datos segment 'data'
msg db 'Hola mundo! $'
datos ends
codiso segment 'code'
assume cs:codiso, ds:datos, ss:rila
main PROC
mov ax,datos
mov ds,ax

mov cx,0 ; contador de nuestro bucle
for:
mov dx,OFFSET msg
mov ah,9
int 20h
inc cx ; incrementamos el contador del bucle
cmp cx,7 ; si el contador no es 7...
jne for ; ...no salgo del bucle

mov ax,2C00h
int 20h
main ENDP
codiso ends
END main

```

16:22

```

int86(0x10, &inress, &outress);
}

int main(){
int tmp;

printf("\nPulsa una tecla... ");
tmp = mi_getchar();
}

```

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

C:\S1>EDITAR.BAT HOLA  
C:\S1>C.BAT HOLA  
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International  
Assembling file: HOLA.asm  
Error messages: None  
Warning messages: None  
Passes: 1  
Remaining memory: 472k  
Turbo Link Version 5.1 Copyright (c) 1992 Borland International  
C:\S1>HOLA.EXE  
Hola mundo! Hola mundo! Hola mundo! Hola mundo! Hola mundo! Hola mundo!  
C:\S1>\_