



Escuela de  
Ingeniería y Arquitectura  
**Universidad** Zaragoza

---

# PRÁCTICA 5 - Ejecución de aplicaciones para arquitecturas desconocidas

GRUPO 3 - Sistemas Legados

---

David Zandundo Fuster (780500)  
Álvaro Andrada Gimeno (795326)  
Rubén Albacete Seren (736650)

<b>Introducción</b>	<b>3</b>
<b>Fichero legado1.bin</b>	<b>3</b>
<b>Fichero legado2.bin</b>	<b>7</b>
<b>Fichero SB.txt</b>	<b>10</b>
<b>Principales dificultades encontradas</b>	<b>13</b>

# Introducción

En esta práctica proporciona dos ficheros binarios, **legado1.bin** y **legado2.bin**, que se han recuperado de un backup antiguo. No se dispone de documentación ni código fuente, solamente se sabe que **legado1.bin** corresponde a una especie de juego 3D y **legado2.bin** es un gestor de stocks. El objetivo es ejecutar las aplicaciones correctamente y que muestren las pantallas que se proporcionan en el guión de la práctica.

La segunda parte de la práctica consiste en conseguir reproducir un fichero de texto (**SB.txt**) diseñado para la aplicación Text Assist. El objetivo es conseguir que podamos volver a escuchar al ordenador cantar esa canción, grabarla, y obtener un fichero mp3 de la misma.

## Fichero legado1.bin

Para analizar el fichero **legado1.bin** se utilizó el comando *'file'*, un comando UNIX que realiza una serie de pruebas (test) para determinar el tipo y formato de un archivo. Este es el resultado de ejecutar el comando:

```
david@DESKTOP-8KA000I:/mnt/c/Users/David/Desktop/practica5_legados$ file legado1.bin
legado1.bin: Macintosh HFS data (bootable) block size: 512, number of blocks: 2874, volume name: Games #3
```

Como se observa en la imagen el fichero corresponde a un volumen HFS de un sistema Macintosh. Para poder abrir el archivo habrá que descargarse un emulador de Macintosh, en nuestro caso se ha decidido utilizar el emulador [Mini vMac](#). Con esto es posible montar el volumen desde el fichero y ejecutar la aplicación.

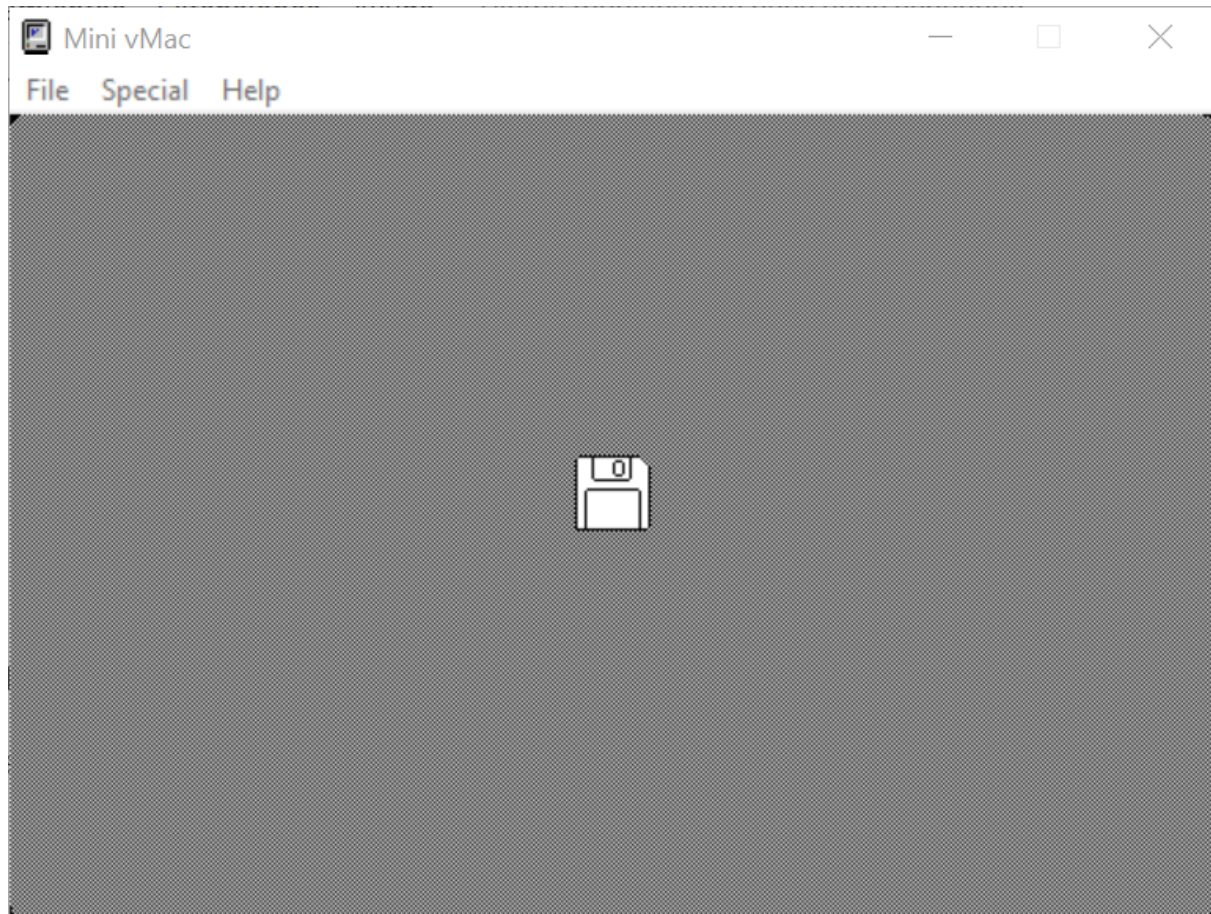
Cuando abrimos por primera vez la aplicación nos aparece la siguiente pantalla:



Como se indica en la imagen, para ejecutar el emulador es necesario conseguir primero la ROM de un Macintosh Plus (hardware emulado por las variaciones estándar del emulador). Para encontrar una ROM válida buscamos una ROM pirata en la web. Encontramos en la primera [página](#) que nos sale, se puede descargar una ROM, nos indican que descarguemos

el fichero '*vmac.rom*', lo renombramos por '*vMac.ROM*' y lo situemos en el mismo directorio donde está el emulador.

Una vez que iniciamos el emulador nos aparece la siguiente nueva pantalla:

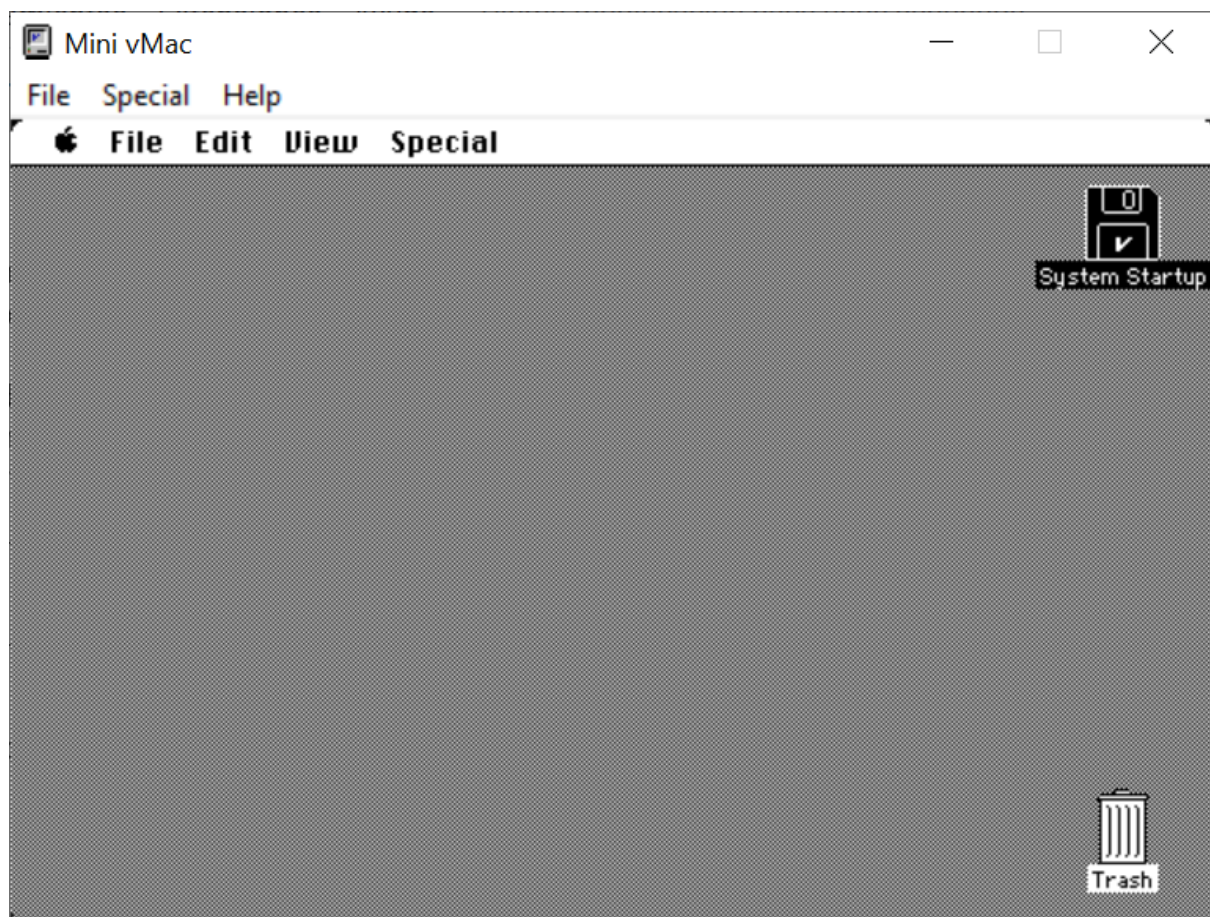


Ahora es necesario contar con una imagen del sistema operativo. *System 6.0.8* está disponible de manera gratuita en la página web de Apple: [disco 1](#), [disco 2](#). Estos ficheros están comprimidos con el software [Stuffit](#).

Para descomprimirlos la página web nos recomienda utilizar el comando [ua608d](#), para ello descargamos el fichero '*ua608d.exe*' de la página web y lo utilizamos de la siguiente manera en la línea de comandos de Windows:

```
ua608d.exe SSW_6.0.8-1.4MB_Disk1of2.sea.bin "System Startup"  
ua608d.exe SSW_6.0.8-1.4MB_Disk2of2.sea.bin "System Additions"
```

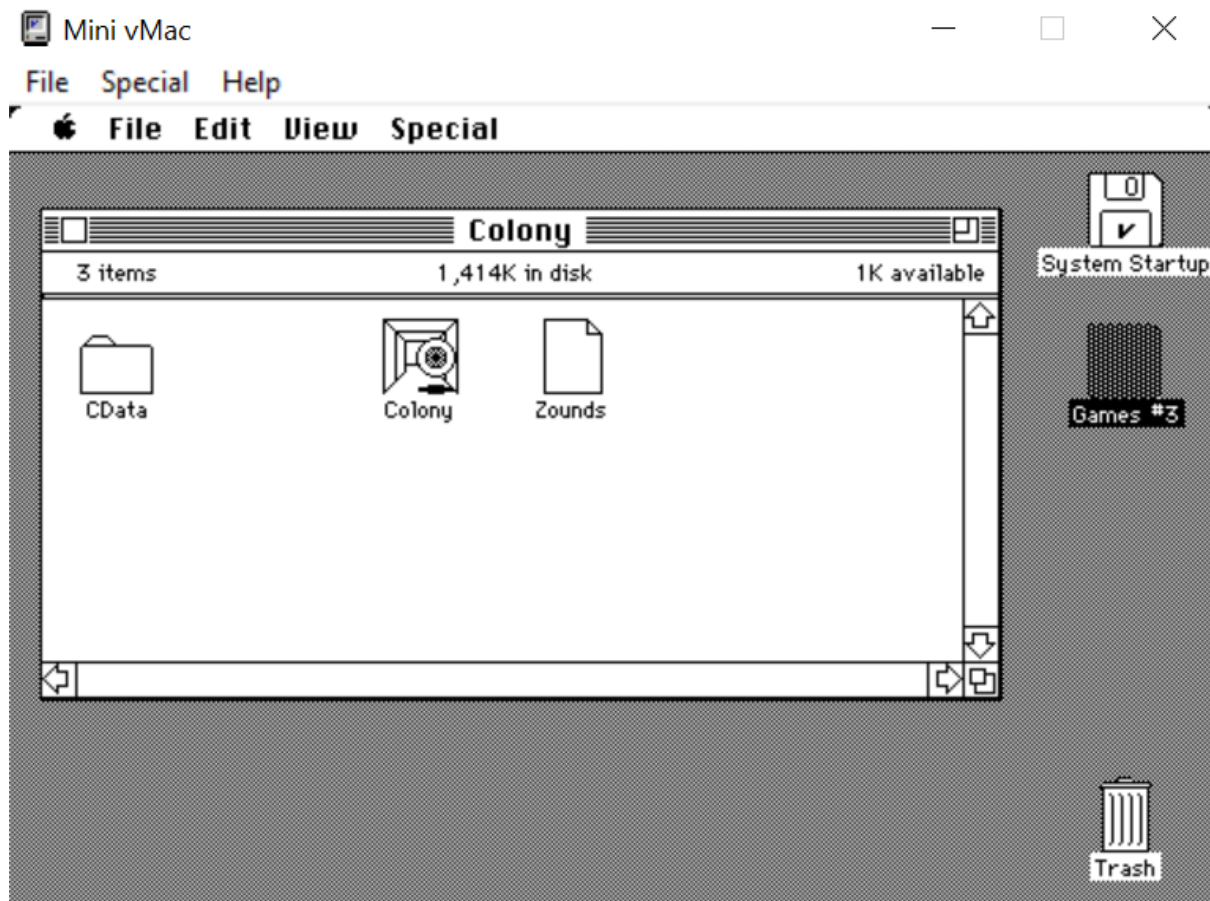
Ahora iniciamos el emulador, cargamos el fichero "*System Startup*" creado anteriormente y nos aparece la siguiente imagen:



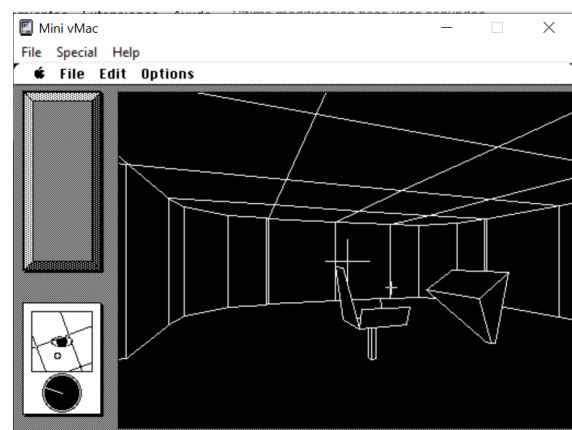
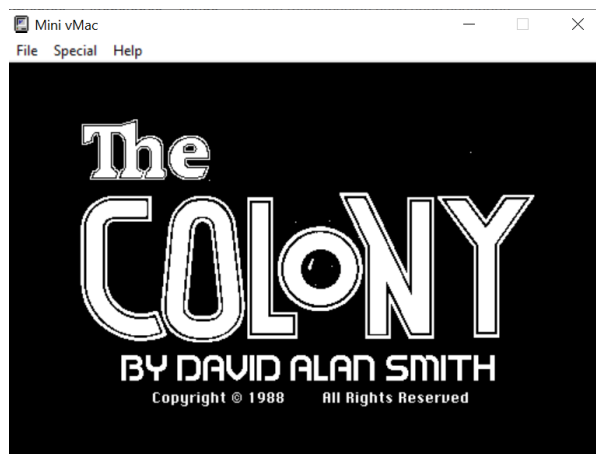
Por último, ya podemos cargar el fichero **legado1.bin**, para ello pulsamos la opción '*File*', luego '*Open Disk Image...*' y por último seleccionamos el fichero '**legado1.bin**' de nuestro ordenador. En la siguiente imagen se muestran los pasos a seguir:



Una vez que hayamos hecho nos aparecerá la siguiente pantalla en el emulador:



Ahora solo tendremos que hacer doble click en el archivo '*Colony*' y se nos ejecutará el juego en 3D, pudiendo jugar con toda la normalidad. A continuación se muestran unas cuantas imágenes del juego:



# Fichero legado2.bin

Para el fichero **legado2.bin** se sigue el mismo procedimiento que el fichero **legado1.bin** y este es el resultado:

```
david@DESKTOP-8KA0001:/mnt/c/Users/David/Desktop/practica5_legados$ file legado2.bin
legado2.bin: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-x.so.2, for GNU/Linux 2.6.15, BuildID[sha1]=0d11cdb7be3800c844625e32520df530e7a5ee6d, not stripped
```

El binario contiene una aplicación compilada para GNU/Linux de 32-bits. Para poder ejecutarlo hay que descargarse librerías de gcc para poder ejecutarlo, ya que nuestra arquitectura es de 64-bits. Para ello se ha descargado [gcc-multilib](#), que es una herramienta para la compilación cruzada, es decir, compilar un programa para ejecutarlo en una arquitectura de procesador diferente, mediante el siguiente comando:

```
sudo apt-get install gcc-multilib
```

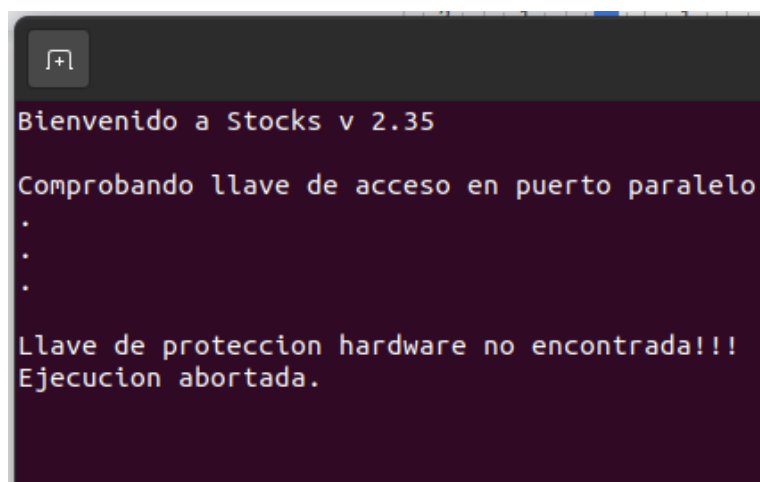
Una vez descargado se ejecuta el programa (./legado2.bin), y nos aparece la siguiente pantalla:



```
Bienvenido a Stocks v 2.35
Comprobando llave de acceso en puerto paralelo
.
.
.

Violación de segmento ('core' generado)
david@david-PortatilMSI:~/Escritorio/p$
```

Se ejecuta el programa pero ocurre un error de 'violación de segmento', para solucionar esto necesitamos ejecutar la aplicación con suficientes permisos para acceder al puerto I/O (CAP\_SYS\_RAWIO). Para ello solo necesitamos ejecutar la aplicación como usuario root del sistema (sudo ./legado2.bin). Cuando lo hacemos aparece la siguiente pantalla:



```
Bienvenido a Stocks v 2.35
Comprobando llave de acceso en puerto paralelo
.
.
.

Llave de proteccion hardware no encontrada!!!
Ejecucion abortada.
```

Por la salida que genera parece que intenta acceder a un dispositivo en el puerto paralelo para buscar una llave de protección hardware. A continuación, se decide ejecutar el programa con [strace](#), una herramienta que muestra las llamadas al sistema que realiza el ejecutable, así como el resultado que devuelve el sistema operativo.

Ejecutamos el siguiente comando y el resultado es el siguiente:

```
sudo strace -y ./legado2.bin > /dev/null
```

```
clone(child_stack=0x17ee6000, flags=CLONE_VM|CLONE_VFORK|SIGCHLD) = 10279
munmap(0xf7ee6000, 36864) = 0
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
close(4<pipe:[163161]>) = 0
fcntl64(3<pipe:[163161]>, F_SETFD, 0) = 0
fstat64(3<pipe:[163161]>, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
read(3<pipe:[163161]>, "-rwxrwx-r-- 1 david david 7907 dl"... , 4096) = 57
close(3<pipe:[163161]>) = 0
wait4(10279, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 10279
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=10279, si_uid=0, si_status=0, si_etime=0, si_stime=0} ---
ioperm(0x378, 0x2, 1) = 0
write(1</dev/null>, "Bienvenido a Stocks v 2.35\n\n7Com"... , 153) = 153
exit_group(1) = ?
+++ exited with 1 +++
david@david-PortatilMSI:~/Escritorio/p$
```

Como se aprecia en la salida de strace, el programa realiza la llamada al sistema [ioperm](#) para activar los bits de permiso de acceso al dispositivo en el puerto de entrada/salida 0x378 (correspondiente al puerto paralelo LPT1), y no realiza ninguna llamada al sistema más antes de mostrar el mensaje de error "Llave de protección hardware no encontrada!!!". Esto significa que se accede a un dispositivo en el puerto paralelo LPT1 para determinar si se tiene acceso o no al programa.

Entonces para determinar si el programa tiene acceso al dispositivo necesitamos un descompilador para tener todas las instrucciones a nivel de ensamblador y ver qué ocurre. Para ello se ha descargado el descompilador de [IDA](#) (Interactive Decompiler) para Linux. Cuando abrimos el programa, le pasamos el fichero binario y buscamos por los comentarios que tiene el programa encontramos el siguiente fragmento de código:

```
.text:08048754 ; ===== S U B R O U T I N E =====
.text:08048754 ; Attributes: bp-based frame
.text:08048754
.text:08048754 inb          proc near          ; CODE XREF: main+A8+p
.text:08048754
.text:08048754 var_14        = word ptr -14h
.text:08048754 var_1         = byte ptr -1
.text:08048754 arg_0         = dword ptr  8
.text:08048754
.text:08048754 |          push    ebp
.text:08048755          mov     ebp, esp
.text:08048757          sub     esp, 14h
.text:0804875A          mov     eax, [ebp+arg_0]
.text:0804875D          mov     [ebp+var_14], ax
.text:08048761          movzx   eax, [ebp+var_14]
.text:08048765          mov     edx, eax
.text:08048767          in      al, dx
.text:08048768          mov     [ebp+var_1], al
.text:0804876B          movzx   eax, [ebp+var_1]
.text:0804876F          leave
.text:08048770          retn
.text:08048770 inb          endp
.text:08048770
```

Subrutina de la función *inb*



```

.text:08048A44      mov     dword ptr [esp+4], 4
.text:08048A4C      mov     dword ptr [esp], 378h
.text:08048A53      call    _ioperm
.text:08048A58      mov     dword ptr [esp], 378h
.text:08048A5F      call    inb
.text:08048A64      cmp     al, 4Dh ; 'M'
.text:08048A66      jz      short loc_8048A91
.text:08048A68      mov     dword ptr [esp], offset aLlaveDeProtecc ; "Llave de proteccion hardware no encontr..."
.text:08048A6F      call    _puts
.text:08048A74      call    beep
.text:08048A79      mov     dword ptr [esp], offset aEjecucionAbort ; "Ejecucion abortada.\n\n\n"
.text:08048A80      call    _puts
.text:08048A85      mov     dword ptr [esp], 1
.text:08048A8C      call    _exit
.text:08048A91      loc_8048A91:
.text:08048A91      ; CODE XREF: main+AF+j
.text:08048A91      mov     dword ptr [esp], offset aLlaveDeProtecc_0 ; "Llave de proteccion encontrada. Ejecuci..."
.text:08048A98      call    _puts
.text:08048A98      call    beep
.text:08048A9D      mov     dword ptr [esp], 3
.text:08048AA2      call    _exit

```

Código ensamblador de la función principal del programa (main)

Después de la llamada al sistema *'ioperm'*, el programa ejecuta una función llamada *'inb'*. Esta función ejecuta la instrucción *'in'* para leer del puerto paralelo y devuelve el byte leído. Después se comprueba que el valor devuelto por *'inb'* sea igual a 0x4Dh. Si esta comprobación tiene éxito, el programa asume que la llave de protección está presente. De lo contrario, aborta la ejecución.

Para hacer creer al programa que la llave de protección está conectada y conseguir que funcione hay dos opciones:

- Crear un dispositivo virtual en el puerto LP1 que devuelva 0x4Dh
- Modificar la condición del programa para que siempre devuelve TRUE y salte siempre a *'loc\_8048A91'*

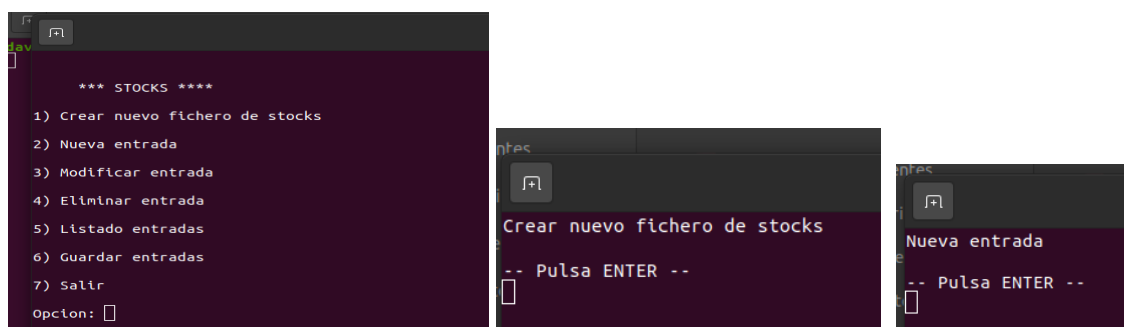
Se ha decidido hacer la segunda opción porque resulta más fácil. Para ello editamos la función *'jz short loc\_8048A91'* para que salte siempre (*'jmp short loc\_8048A91'*), quedando las instrucciones de la siguiente manera:

```

.text:08048A4C      mov     dword ptr [esp], 378h
.text:08048A53      call    _ioperm
.text:08048A58      mov     dword ptr [esp], 378h
.text:08048A5F      call    inb
.text:08048A64      cmp     al, 4Dh ; 'M'
.text:08048A66      jmp     short loc_8048A91
.text:08048A68      ; -----
.text:08048A68      mov     dword ptr [esp], offset aLlaveDeProtecc ; "Llave de proteccion hardware no encontr..."
.text:08048A6F      call    _puts
.text:08048A74      call    beep
.text:08048A79      mov     dword ptr [esp], offset aEjecucionAbort ; "Ejecucion abortada.\n\n\n"
.text:08048A80      call    _puts
.text:08048A85      mov     dword ptr [esp], 1
.text:08048A8C      call    _exit
.text:08048A91      loc_8048A91:
.text:08048A91      ; CODE XREF: main+AF+j
.text:08048A91      mov     dword ptr [esp], offset aLlaveDeProtecc_0 ; "Llave de proteccion encontrada. Ejecuci..."
.text:08048A98      call    _puts

```

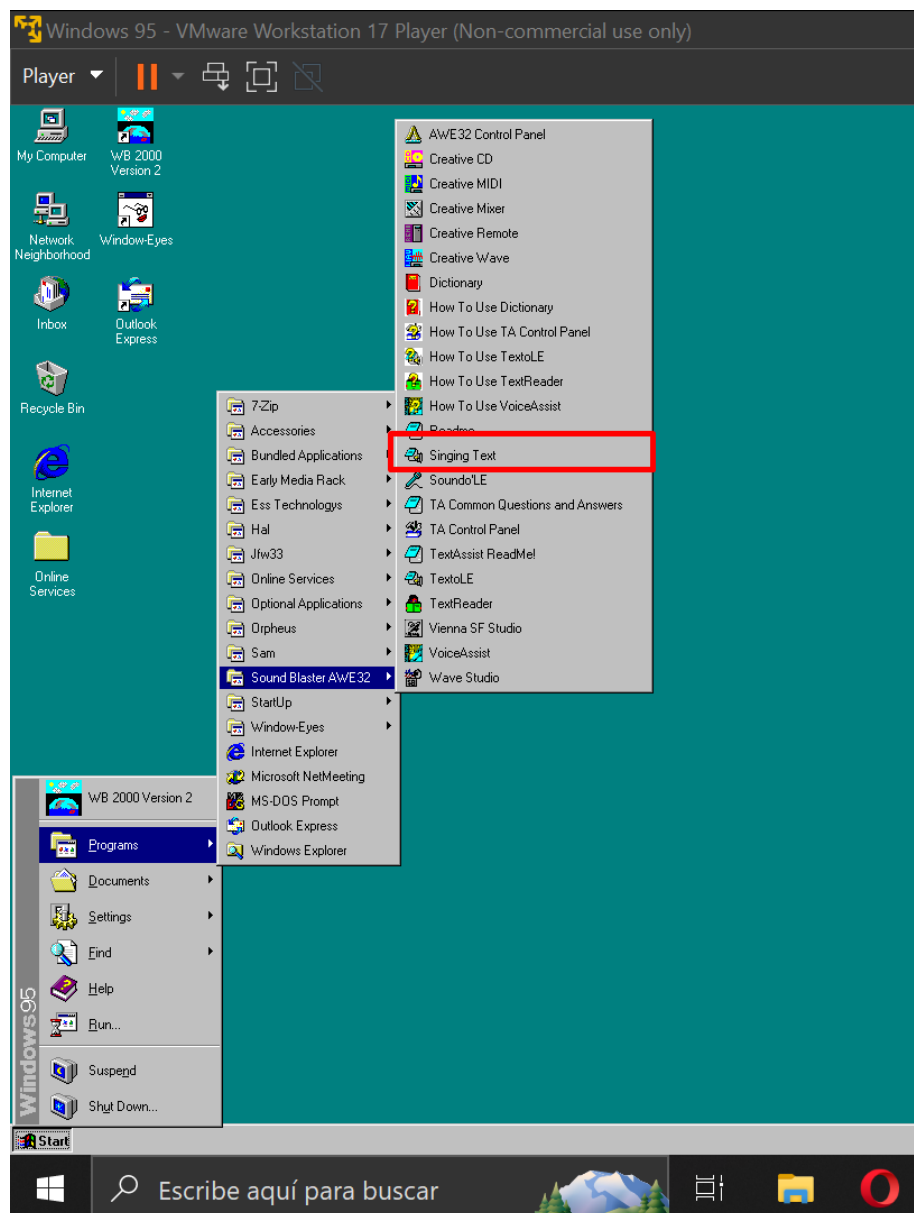
Por último guardamos los cambios, ya que IDA permite guardar los cambios en el mismo fichero binario que se edita, volvemos a ejecutar de nuevo (`sudo ./legado2.bin`) y este es el resultado:



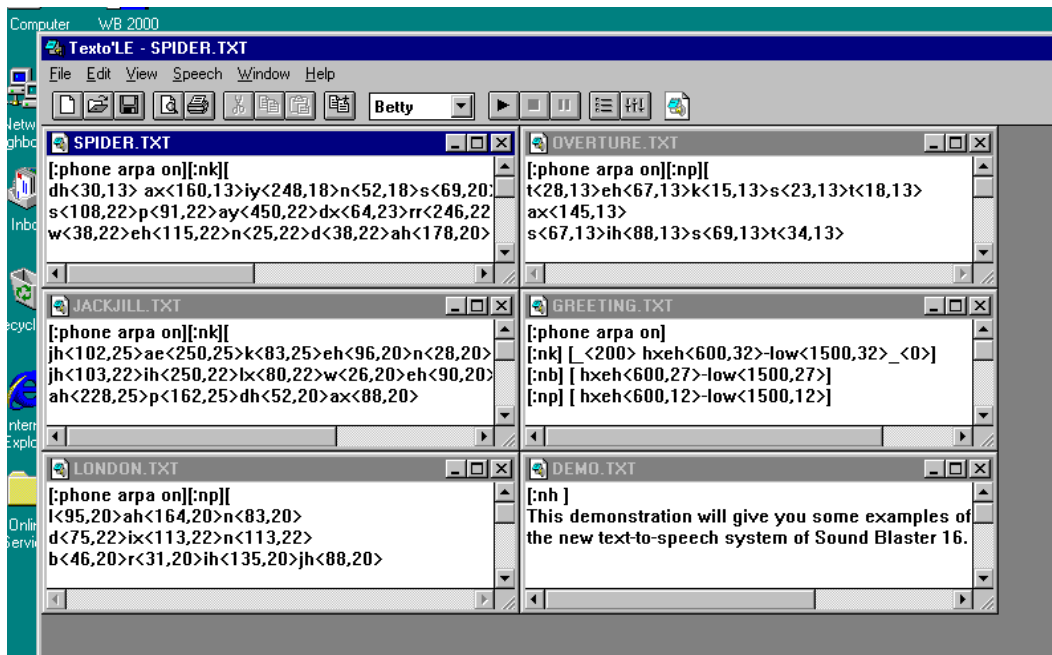
# Fichero SB.txt

Para completar este apartado, primero se empezó por buscar información sobre la aplicación 'Text Assist' de las tarjetas de sonido [Creative Sound Blaster 16](#). Se encontró que la aplicación estaba disponible para Windows 95, por lo que se decidió buscar alguna distribución de Windows 95 que contuviera los drivers y el software necesario para esta aplicación.

Para esta tarea nos ha resultado muy útil la herramienta de [WayMachine](#) de [archive.org](#), ya que pudimos encontrar una llamada página de <http://grossgang.com/> (actualmente no disponible en la web) que nos puede proveer la imagen de diferentes SO para VMware, más concretamente nosotros queremos la de [Windows 95](#) para VMware. Una vez que nos descargamos la imagen y configuramos el VMware, iniciamos el Windows 95 y buscamos la aplicación.

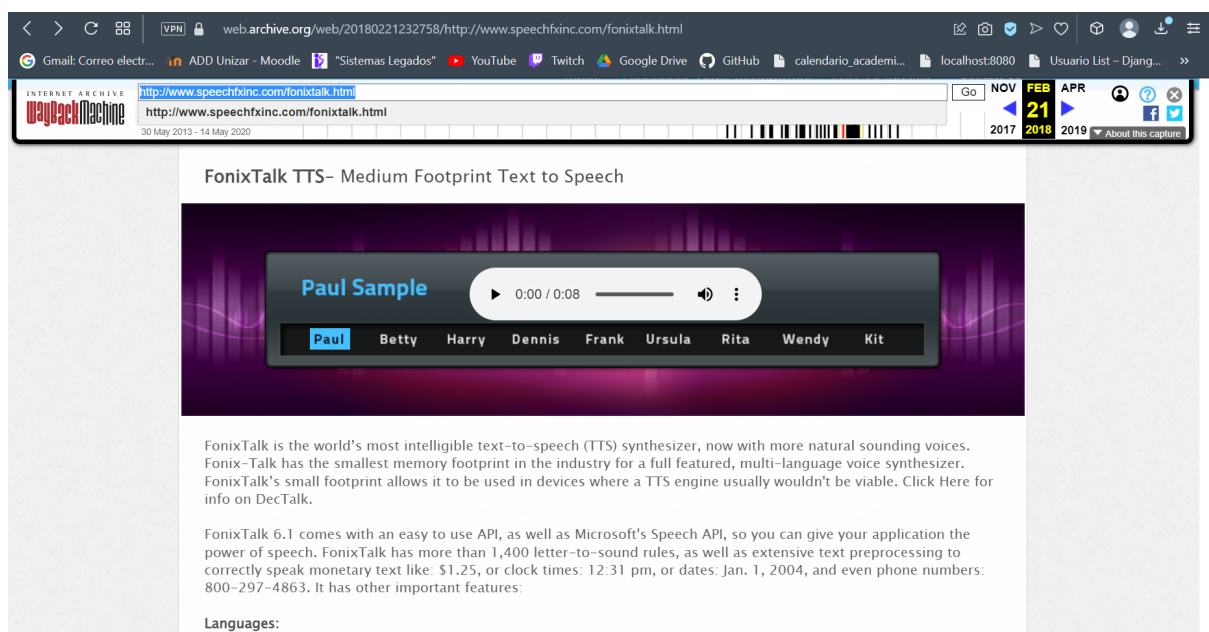


Cuando la abrimos la aplicación nos aparece la siguiente pantalla:

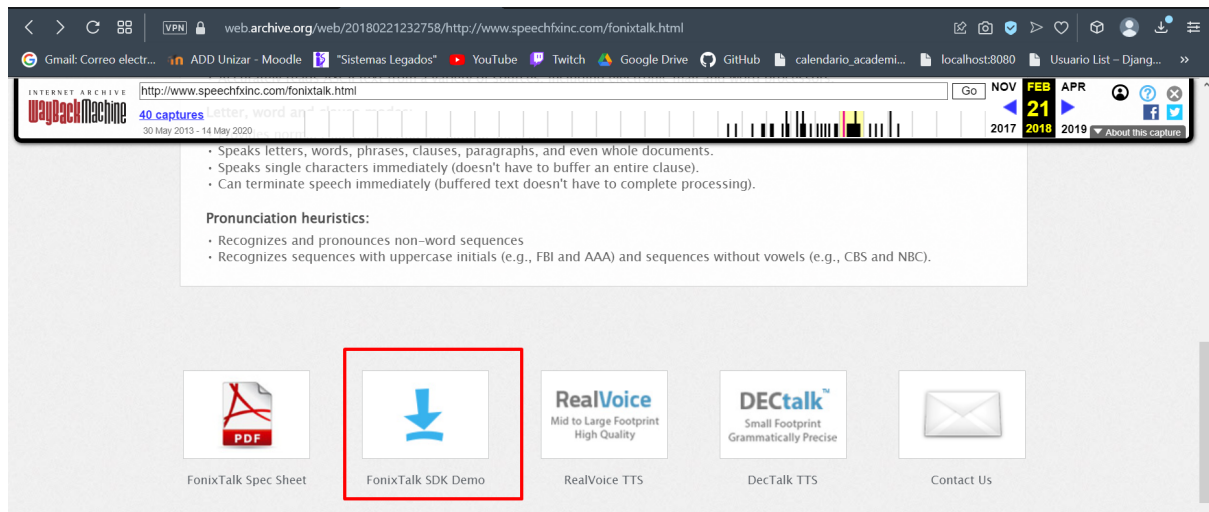


Como se observa es la aplicación que queremos, incluso podemos escuchar unas canciones de ejemplo, pero hay un problema, la aplicación está en inglés, y cuando le escribimos la canción de 'SB.txt' no sabe cantar y nos da varios errores ya que la fonética del vasco al inglés es muy diferente. Por ello intentamos buscar otra imagen que contenga los mismos drivers y software pero en español, pero no pudimos encontrar ninguna.

Por ello buscamos aplicaciones que su lógica sea parecida a la aplicación de 'Text Assist'. Encontramos una aplicación llamada [FonixTalk TTS](http://www.speechfxinc.com/fonixtalk.html) que era prácticamente igual a 'Text Assist' y además tenía el castellano, pero su página estaba caída, y decidimos recurrir otra vez a [WayMachine](http://www.waybackmachine.org/).

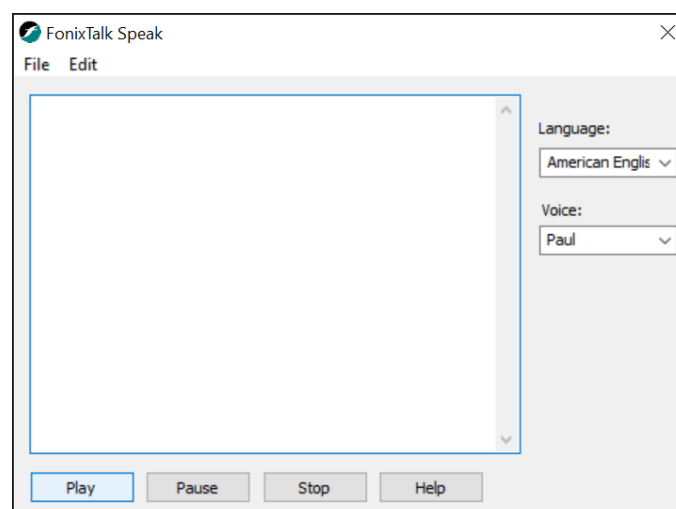


Además nos ofrecía una opción para descargar el software.

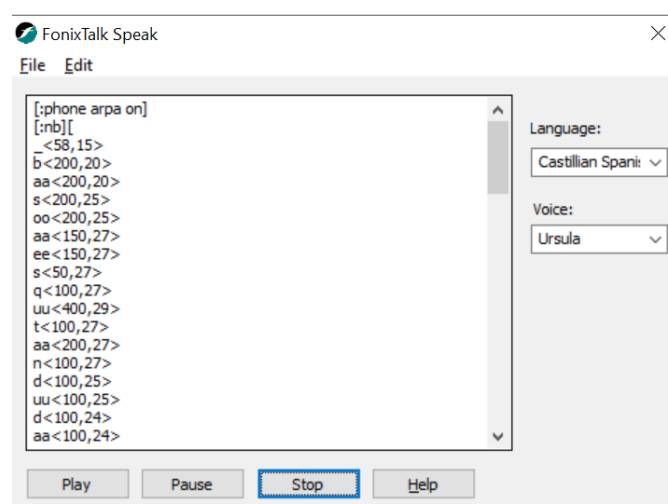


Una vez descargado, entre muchos otros, obtenemos un archivo llamado 'Speak\_CPP.exe'.

Lo ejecutamos y nos aparece la siguiente pantalla:



Como se observa nos aparece una opción de cambiar el idioma, seleccionamos el español y copiamos el texto de 'SB.txt':



Por último, le damos al botón de 'Play' y ya podemos disfrutar de la canción.

## Principales dificultades encontradas

La principal dificultad encontrada ha sido encontrar todo el software requerido para la realización de la práctica, debido a que la mayoría no se encontraba por la web y se ha tenido que utilizar la herramienta de [WayMachine](#), así como la poca información que se encontraba por la web para solucionar los problemas que nos surgían.