Práctica 5: Ejecución de aplicaciones desconocidas/no disponibles

Sistemas Legados

Pedro Allué Tamargo (758267) Juan José Tambo Tambo (755742)
Jesús Villacampa Sagaste (755739)

2 de febrero de 2021

Índice

1. Introducción

En esta práctica se va a proceder a ejecutar aplicaciones desconocidas de las cuales no se tiene ninguna información. Para esta práctica se han proporcionado dos programas legados. Estos programas se corresponden con un juego y con un sistema de control de inventario.
También se debe reproducir una canción en formato *text assist* para las tarjetas de sonido *Sound Blaster*. Este programa no está disponible y habrá que realizar tareas extra para poder conseguir el objetivo.

2. Esfuerzos invertidos

- Pedro Allué Tamargo: Ejecución del *legado1.bin*, *legado2.bin*, modificación del código del *legado2.bin* y apoyo a la recuperación del *software* necesario para reproducir la canción. Memoria de los programas legados.
  - Tiempo invertido: 8 horas.
- Juan José Tambo Tambo: Ejecución del *legado2.bin* y y apoyo a la recuperación del *software* necesario para reproducir la canción.
  - Tiempo invertido: 4 horas.
- Jesús Villacampa Sagaste: Ejecución del *legado2.bin*, modificación del código del *legado2.bin*, creación del entorno para reproducir la canción y su posterior grabación. Memoria de la parte de la canción.
  - Tiempo invertido: 8 horas.

3. Aplicaciones Legadas

3.1. Aplicación *legado1.bin*

El primer legado se puede abrir con un editor de texto plano y se puede observar la cadena *The Polony David A. Smith* y *1988*. Realizando una búsqueda en Internet se puede observar que se corresponde con el juego *"The Colony"*[1]. Se corresponde con la captura de pantalla dada en el enunciado.
Utilizando el comando `file legado1.bin` se puede observar la siguiente salida:

```
→  BinariosDesconocidos git:(master) file legado1.bin
legado1.bin: Macintosh HFS data (bootable) block size: 512, number of blocks: 2874, volume name: Games #3
```

Figura 1: Captura de pantalla de la salida del comando `file legado1.bin`

Se puede observar que ser corresponde con un fichero de sistema de ficheros de un *Macinstosh*. Por lo tanto el programa legado se corresponde con la versión de *The Colony* para *Macintosh*.

Se va a utilizar un emulador *Macintosh* para ejecutar el programa. El emulador utilizado será *mini vMac*. Para la preparación del entorno de emulación se han seguido las indicaciones encontradas en este tutorial[2].
Una vez instalado el sistema operativo de *Macintosh* en el emulador se va a proceder a introducir el fichero *legado1.bin*. Para ello se arrastrará el fichero hasta la ventana del emulador. Una vez cargado el sistema de ficheros aparecerá un icono en el escritorio (Figura 4). Dentro de este sistema de ficheros se puede encontrar el fichero *Colony* (Figura 5). Si lo ejecutamos se cargará el juego y se verá la imagen de la Figura 6.

[1] `https://en.wikipedia.org/wiki/The_Colony_(video_game)`
[2] `https://www.emaculation.com/doku.php/mini_vmac_setup`

3.2. Aplicación *legado2.bin*

El segundo legado tras abrirlo con un editor de texto plano se puede observar la cadena de texto *ELF* al inicio del fichero. Si se ejecuta el comando `file legado2.bin` se puede observar la siguiente salida:

```
→  BinariosDesconocidos git:(master) file legado2.bin
legado2.bin: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6
.15, BuildID[sha1]=0d11cdb7be3800c844625e32520df530e7a5ee6d, not stripped
```

Figura 2: Captura de pantalla de la salida del comando `file legado2.bin`

Se puede observar que es un binario de 32 bits compilado para el *kernel* de Linux 2.6.15 y para un procesador *Intel 80386*. Al ser de arquitectura x86 se podrá ejecutar sobre un procesador de 64 bits pero se va a utilizar una máquina virtual *Ubuntu 16.04* de 32 bits que es la última que utilizaba el kernel para el que estaba compilada la máquina.

Si se inspecciona el fichero se puede observar que se nombra a una llave de protección *Hardware*. Esto significa que el programa tiene una interacción con un elemento *Hardware* externo que impedirá la ejecución del programa con el resultado esperado. Para ejecutar el programa se ha utilizado el comando `sudo ./legado2.bin`. El resultado de la ejecución puede observarse en la Figura 3.

```
Bienvenido a Stocks v 2.35

Comprobando llave de acceso en puerto paralelo
.
.
.

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

Figura 3: Captura de pantalla de la ejecución de `legado2.bin` sin modificación de llave hardware

Se puede observar que el fichero no se ejecuta correctamente debido a la restricción de la llave de protección *Hardware*. Por lo tanto se va a proceder a modificar el código de la aplicación legada. Para ello se va a utilizar el programa *Ghidra*[3]. Tras abrir el fichero como parte de un proyecto se va a inspeccionar su código con la opción de 'Analizar' que proporciona *Ghidra* y se puede observar su función `main` (Figura 9). accediendo a él desde *Symbol Tree -> Functions*.

Para modificar el flujo de control del programa se ha modificado la instrucción `JZ` (Figura 11) que salta a una dirección de memoria (instrucción `puts("Llave de proteccion encontrada. Ejecucion permitida.");`) si la llave *hardware* se había introducido. Se ha modificado por una instrucción de salto incondicional `JMP` (Figura 12), por lo que la claúsula *if* de Figura 9 nunca se ejcutará y el código resultante será el de la Figura 10

Por lo tanto, ejecutando el programa sobre la plataforma anterior se obtiene un error de segmentación. Este error se corresponde con un error de la herramienta[4]. Para corregir esto se ha utilizado un editor hexadecimal [5] sobre el fichero original. La solución es simple debido a que se conoce el contenido del programa y su codificación en hexadecimal. Se ha buscado la instrucción correspondiente al *JZ* (Figura 14) y se ha modificado por un *JMP* (Figura 15). Si se ejecuta otra vez el programa legado con el comando `sudo ./legado2.bin` se obtiene lo mostrado en la Figura 13.

[3] `https://ghidra-sre.org/`
[4] `https://reverseengineering.stackexchange.com/questions/25427/segmentation-fault-after-export-binary-file-in-ghidra-even-without`
[5] `https://mh-nexus.de/en/hxd/`

4. Canción del fichero

Se debe conseguir un archivo en formato *mp3* la canción que se almacena en el fichero proporcionado con la práctica, *cancion.txt*. Este fichero de texto describe una canción en euskera para la aplicación "Text Assist", incluida en el software distribuido junto a las tarjetas de sonido Creative Sound Blaster 16.

Para obtener la información del sistema operativo en el que se puede instalar la tarjeta de sonido Creative Sound Blaster 16 se observa el ejemplo proporcionado en el guión de la práctica y se concluye que es la tarjeta de sonido del sistema operativo *Windows 95*.

Para crear una máquina virtual de *Windows 95* se sigue paso a paso el siguiente tutorial: `https://www.youtube.com/watch?v=KbLDteibZmA`. En este se crea y configura una máquina Virtual para VirtualBox adecuadamente. Una vez disponible la máquina virtual, se necesita el programa *Text Assist* para reproducir la canción. En el vídeo de ejemplo se puede observar como la tarjeta de sonido es *Sound Blaster AWE64* por lo que se busca en *https://archive.org/details/software* una ISO que correspondan a los drivers de esa tarjeta. Después de varias descargas y pruebas de diferentes ISOs se consigue obtener una ISO que contiene el programa buscado y es la siguiente: `https://archive.org/details/soundblasterawe64iso`.

Una vez se tiene la *ISO*, se introduce como una unidad óptica en la máquina virtual y se descargan los programas. A continuación, con el programa ya disponible en la máquina (Figura 7), se necesita el archivo *cancion.txt* en la máquina virtual. Se ha intentado meterla a la máquina virtual mediante el sistema de carpeta compartida de VirtualBox, pero no ha sido posible. Finalmente se ha optado por crear una *ISO* del archivo *cancion.txt* con el programa *UltraISO*, y que se introduce a la máquina virtual como unidad óptica, donde ya es posible descargarla localmente.

Con el programa y la canción, ya es posible reproducirla (Figura 8) y modificar la voz y distintos parámetros en función de como se desee que suene.

Para grabar la canción en un archivo de extensión *mp3* se conecta un cable *minijack-minijack* a la entrada y a la salida del ordenador, y una vez redirigida la salida (el audio de la canción) a la entrada de audio, se graba con *Audacity* y se obtiene la canción directamente exportada como *mp3* desde el programa.

5. Conclusiones

Tras la realización de la práctica el equipo de trabajo se ha dado cuenta de la importancia de los emuladores y de preservar el software legado. Gracias al emulador de *Macintosh* se ha podido ejecutar el primer programa legado. El segundo programa legado tenía una contramedida *Hardware* pero gracias a la charla de *Miguel Ángel Horna* (alias *"El Semi"*) se pudo eliminar utilizando una herramienta de desensamblado de código. Con esta herramienta se pudo conseguir alterar el código del segundo programa legado para saltarse la contramedida de la llave *Hardware*. Por último, también es importante preservar el software legado. Gracias a la página de *archive.org*[6] se ha podido obtener una copia del *CD* de instalación del *software* necesario para reproducir la canción adjunta al enunciado.

[6] `https://archive.org/`

Anexo 1: Figuras

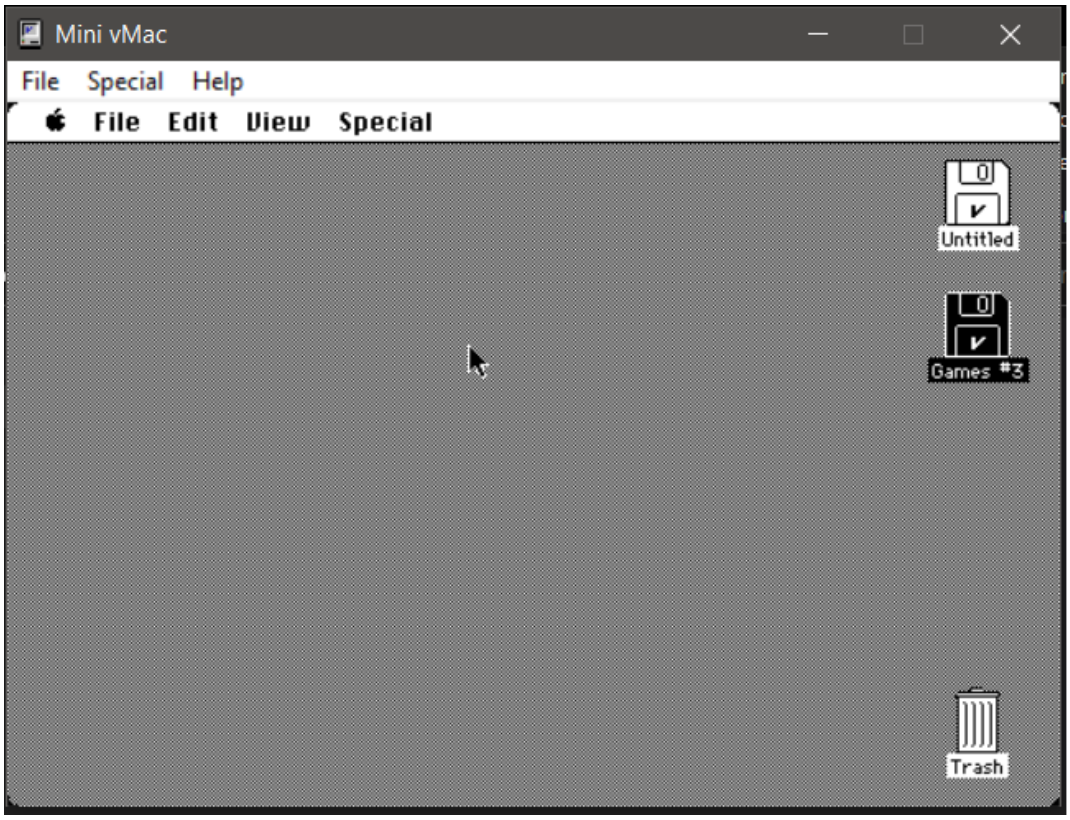

Figura 4: Captura de pantalla del escritorio del emulador *Mini vMac*

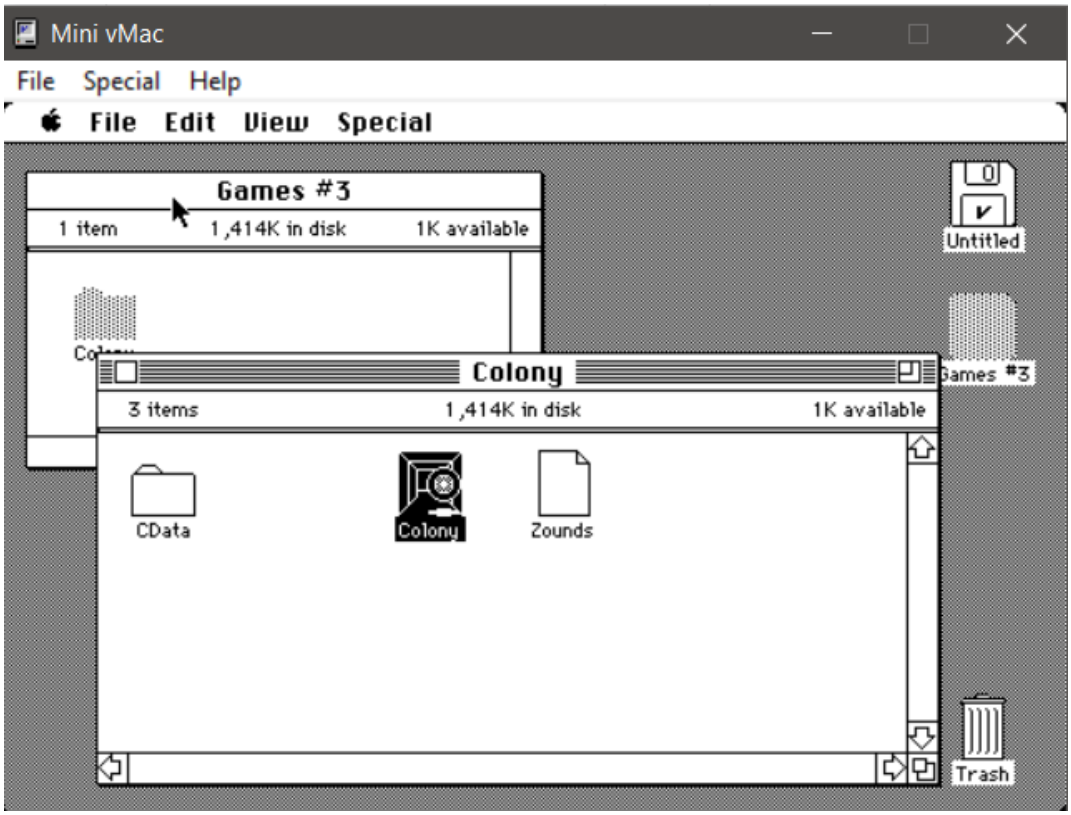

Figura 5: Captura de pantalla del conetenido de *legado1.bin* en el emulador *Mini vMac*

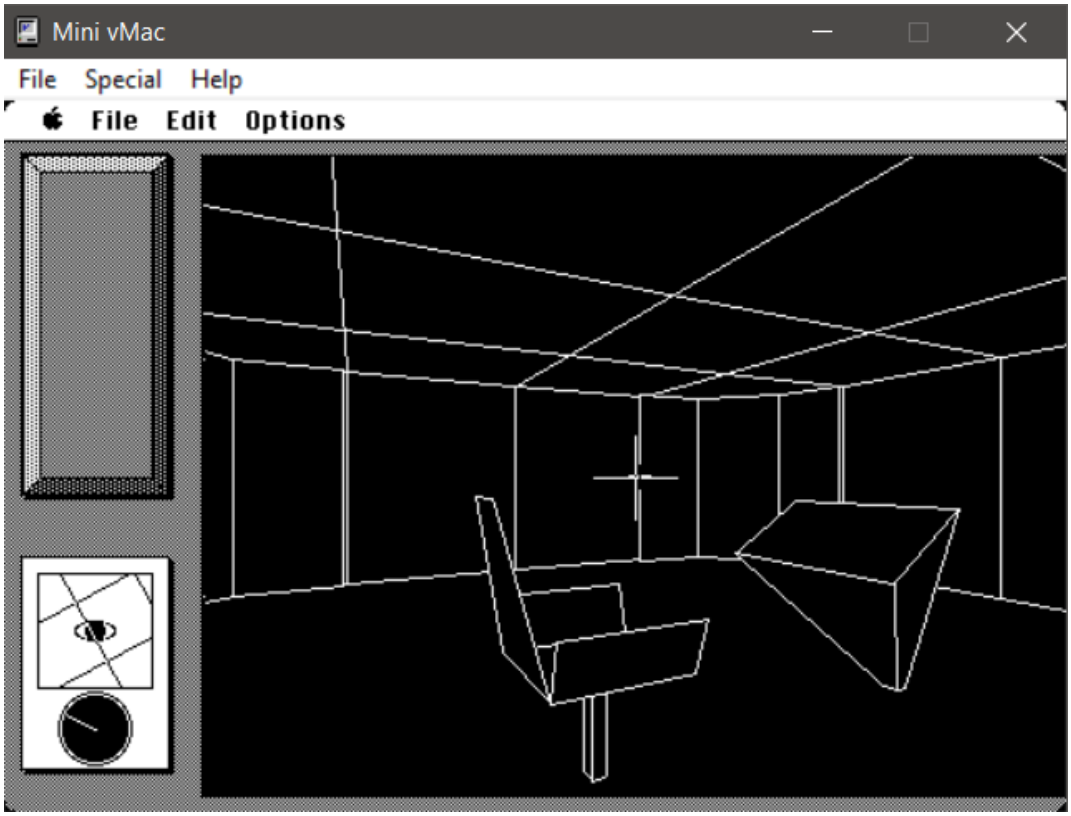

Figura 6: Captura de pantalla del juego *legado1.bin* en el emulador *Mini vMac*

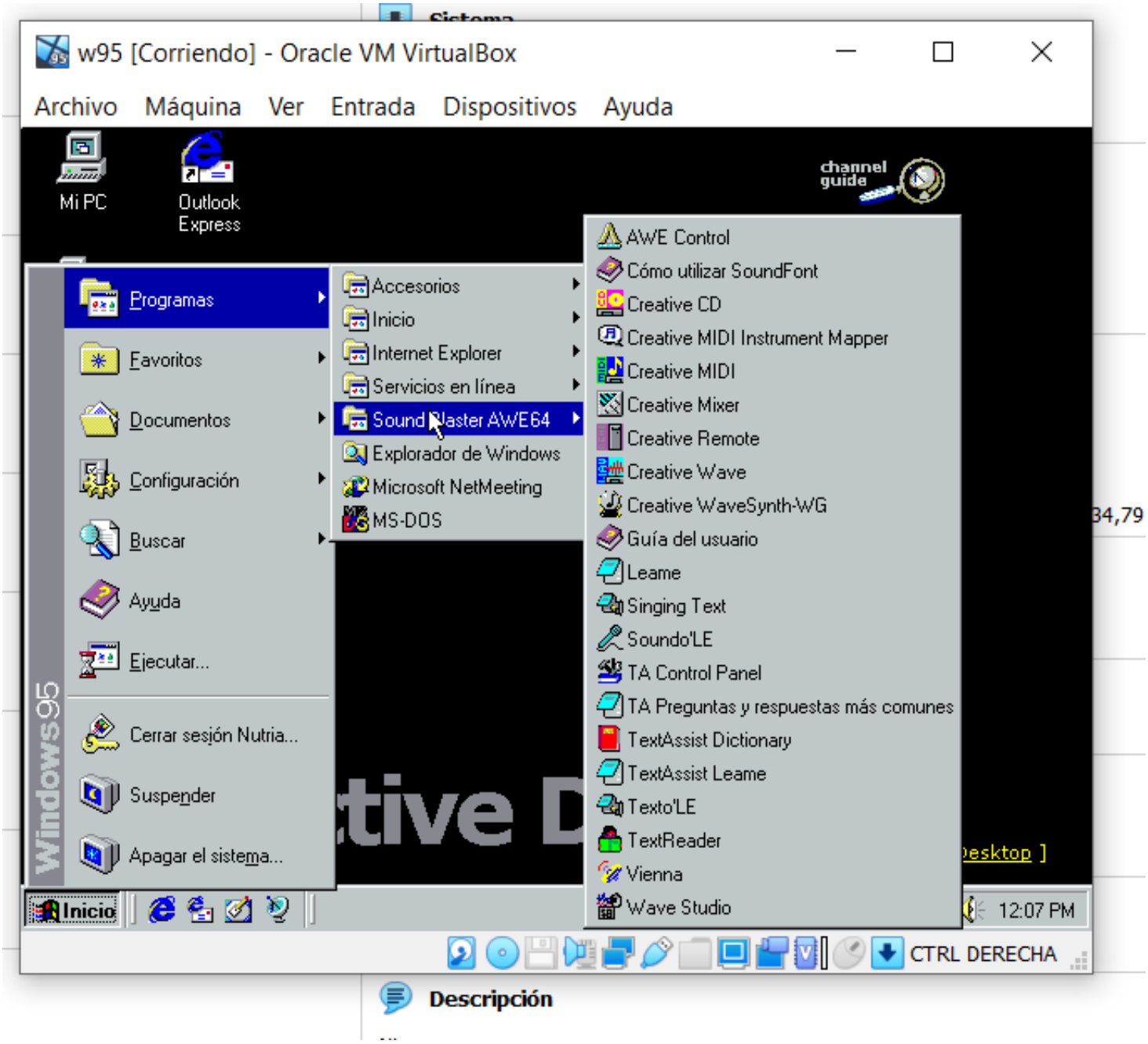

Figura 7: Máquina virtual con los programas de Sound Blaster AWE64

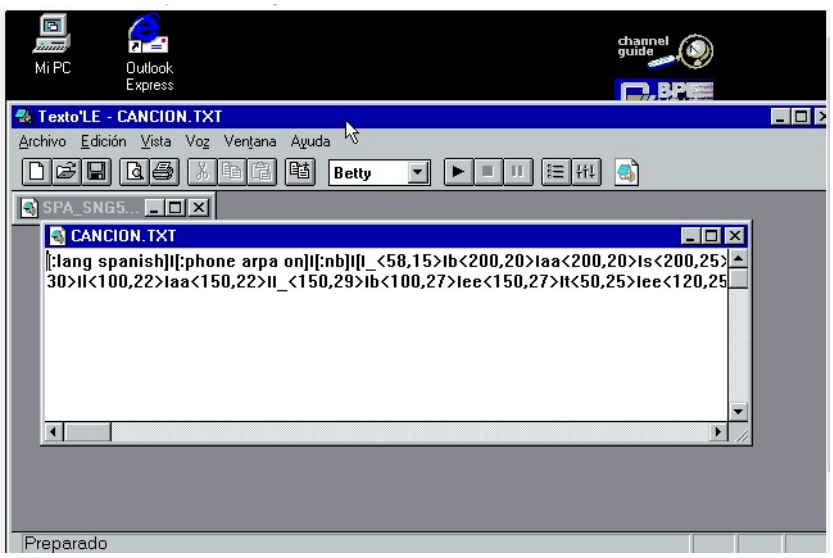

Figura 8: Canción reproducida por Text Assist

```
ioperm(0x378,2,1);
cVar1 = inb(0x378);
if (cVar1 != 'M') {
  puts("Llave de proteccion hardware no encontrada!!!");
  beep();
  puts("Ejecucion abortada.\n\n\n");
                    /* WARNING: Subroutine does not return */
  exit(1);
}
puts("Llave de proteccion encontrada. Ejecucion permitida.");
```

Figura 9: Captura de pantalla del main de `legado2.bin` abierto con *Ghidra*

```
undefined4 main(undefined4 param_1,undefined4 *param_2)

{
  int local_18;
  undefined4 local_14 [4];

  system("clear");
  puts("Bienvenido a Stocks v 2.35\n");
  beep();
  sleep(1);
  puts("Comprobando llave de acceso en puerto paralelo");
  local_18 = 1;
  while (local_18 < 4) {
    puts(".");
    sleep(1);
    local_18 = local_18 + 1;
  }
  putchar(10);
  compruebaPermisos(*param_2);
  ioperm(0x378,2,1);
  inb(0x378);
  puts("Llave de proteccion encontrada. Ejecucion permitida.");
  beep();
  sleep(3);
  do {
    system("clear");
    puts("\n\n     *** STOCKS ****\n");
    puts("1) Crear nuevo fichero de stocks\n");
```

Figura 10: Captura de pantalla del main de `legado2.bin` abierto con *Ghidra* sin la comprobación de la llave hardware

```
08048a64 3c 4d          CMP         AL,0x4d
08048a66 74 29          JZ          LAB_08048a91
08048a68 c7 04 24       MOV         dword ptr [ESP]=>local_30,s_Llave_de_protecci
         a4 8d 04 08
08048a6f e8 fc fb       CALL        puts
         ff ff
08048a74 e8 2a ff       CALL        beep
         ff ff
08048a79 c7 04 24       MOV         dword ptr [ESP]=>local_30,s_Ejecucion_abortac
         d2 8d 04 08
08048a80 e8 eb fb       CALL        puts
         ff ff
08048a85 c7 04 24       MOV         dword ptr [ESP]=>local_30,0x1
         01 00 00 00
08048a8c e8 ef fb       CALL        exit
         ff ff
                     -- Flow Override: CALL_RETURN (CALL_TERMINATOR)

                     LAB_08048a91                                    XREF[1]:
08048a91 c7 04 24       MOV         dword ptr [ESP]=>local_30,s_Llave_de_protecci
```

Figura 11: Código del main en ensamblador con la instrucción JZ

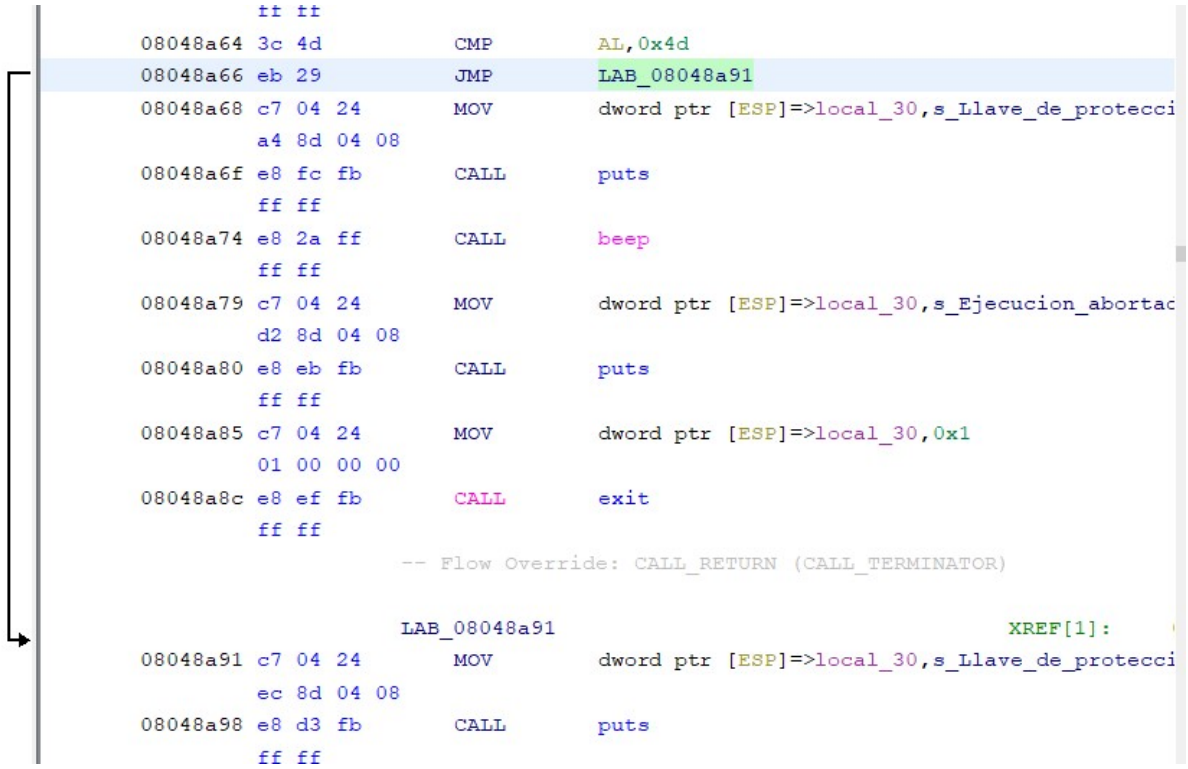

Figura 12: Código del main en ensamblador con la instrucción JMP

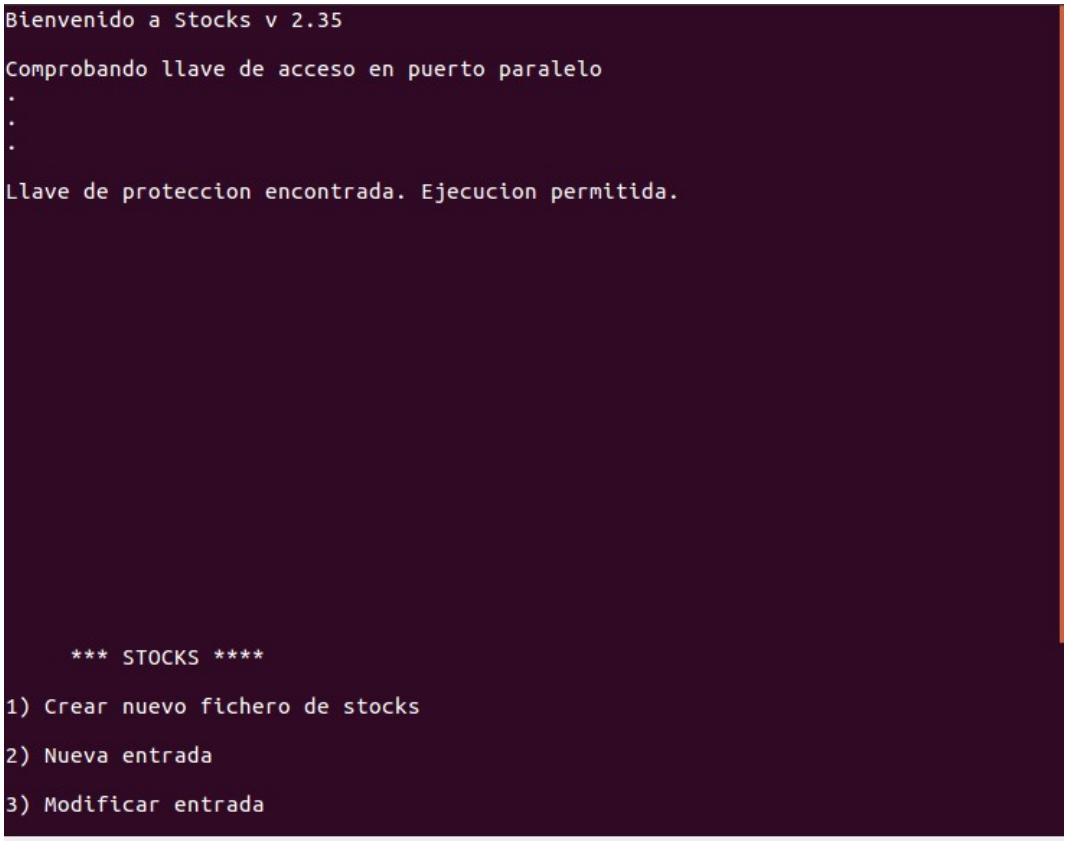

Figura 13: Aplicación *legado2.bin* con el cambio de código de la llave HW

Figura 14: Código hexadecimal con intrucción JZ

Figura 15: Código hexadecimal con intrucción JMP