Desarrollo de un Programa en Ensamblador para el Aprendizaje del Alfabeto Fonético

Miguel David Sánchez Sánchez
Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Heredia, Costa Rica
miguelsanchez712000@estudiantec.cr

***Abstract*—Este documento describe el desarrollo de un bootloader en ensamblador x86 que permite a los usuarios practicar el alfabeto fonético militar mediante un sistema de evaluación automática. Se detallan las herramientas utilizadas, las estructuras de datos implementadas, las instrucciones de ejecución y un análisis de las actividades realizadas durante el desarrollo. Además, se explica cómo el programa está diseñado para funcionar como el sistema de arranque de una computadora.**

***Index Terms*—Bootloader, Ensablador x86, BIOS Interrupts, QEMU, NASM, Alfabeto Fonético**

I. Introducción

El objetivo de esta tarea es implementar un bootloader en ensamblador x86 que, ejecutado en un entorno de bajo nivel, permita a los usuarios practicar la escritura del alfabeto fonético. El sistema selecciona una letra aleatoria y solicita al usuario que la deletree correctamente. Se proporciona retroalimentación basada en la respuesta ingresada. Adicionalmente, se busca que el código funcione como un sistema de arranque para una computadora, cargándose directamente desde el sector de arranque de un disco.

II. Ambiente de Desarrollo

Para la implementación se utilizaron las siguientes herramientas:

- **NASM**: Ensamblador utilizado para compilar el código fuente.
- **QEMU**: Emulador utilizado para ejecutar el programa como un sistema de arranque.
- **GNU/Linux (Ubuntu)**: Sistema operativo donde se desarrolló la tarea.
- **Editor de texto (VS Code, Vim)**: Para la escritura y edición del código.
- **GitHub**: Plataforma utilizada para el control de versiones y almacenamiento del código.

El código fuente del proyecto está disponible en el siguiente repositorio de GitHub:

```
https://github.com/miguel712000/booting-radio-spell
```

III. Estructuras de Datos y Funciones

Las principales estructuras utilizadas en el código son:

- **Tabla de Letras**: Contiene los caracteres del alfabeto.
- **Tabla de Palabras Fonéticas**: Almacena los nombres fonéticos asociados a cada letra.
- **Buffer de Entrada**: Almacena la respuesta ingresada por el usuario.

Las funciones principales incluyen:

- **seleccionar_letra**: Elige una letra aleatoria y su correspondiente palabra fonética.
- **print_puntaje**: Imprime el puntaje en la pantalla.
- **leer_teclado**: Captura la entrada del usuario.
- **comparar**: Verifica si la entrada coincide con la respuesta esperada.
- **print_correcto / print_incorrecto**: Muestran mensajes de validación de la respuesta.

IV. Instrucciones de Ejecución

Para compilar y ejecutar el programa como un bootloader:

```
$ nasm -f bin boot.asm -o boot.bin
$ qemu-system-x86_64 -drive format=raw,file=
    boot.bin
```

Para probar el sistema en una unidad USB:

```
$ sudo dd if=boot.bin of=/dev/sdX bs=512 count
    =1
```

Nota: Reemplazar `/dev/sdX` por el identificador correcto de la unidad USB.

Presionar ESC en cualquier momento cerrará el programa.

V. Actividades Realizadas

Fecha	Actividad	Horas
10/03/2024	Configuración de entorno y prueba en QEMU	3
11/03/2024	Implementación de selección de letras	4
12/03/2024	Manejo de entrada de usuario y comparación	5
13/03/2024	Depuración y solución de errores	6
14/03/2024	Configuración como bootloader y pruebas en USB	5
15/03/2024	Documentación y revisión final	4

TABLE I
Registro de actividades y tiempos de desarrollo.

VI. Autoevaluación

El programa cumple con los requerimientos propuestos, permitiendo la selección aleatoria de letras y validación de respuestas. Además, se logró implementar correctamente como un bootloader, cargándose en el sector de arranque de un disco y ejecutándose en QEMU sin sistema operativo.

Sin embargo, durante las pruebas en hardware real, se identificó que algunas configuraciones de BIOS modernas no permiten la ejecución de código de arranque en modo MBR sin ajustes específicos. Se intentó habilitar el modo Legacy Boot y desactivar Secure Boot, pero la incompatibilidad persistió en algunas configuraciones de hardware. Se realizaron múltiples intentos de escritura del MBR y verificación del contenido con 'hexdump', confirmando que el código estaba correctamente almacenado, pero ciertas restricciones del BIOS impidieron su ejecución.

El control de versiones fue gestionado en GitHub, permitiendo un seguimiento de los cambios y mejoras a lo largo del desarrollo.

En la rúbrica de evaluación, el proyecto recibiría una calificación alta en funcionalidad y documentación, con margen de mejora en optimización del código y compatibilidad con hardware moderno.

VII. Lecciones Aprendidas

- La importancia de verificar el rango de valores en cálculos aleatorios.
- Manejo eficiente de interrupciones de BIOS para entrada y salida de datos.
- Conceptos fundamentales sobre el arranque de sistemas y el uso de bootloaders.
- Limitaciones de hardware y configuraciones de BIOS modernas en el arranque de MBR.

VIII. Bibliografía

References

[1] NASM Documentation, "Netwide Assembler Manual", 2023.
[2] QEMU Official Documentation, 2023.
[3] Tanenbaum, A. "Structured Computer Organization", Pearson, 2012.