Tarea 2: Frogger

Andrés José Ramirez Ortega

April 2021

1 Introducción

Frogger es un juego de arcade desarrollado por Konami y originalmente publicado por Sega. El objetivo del juego es guiar a las ranas a sus hogares, una por una, cruzando carreteras y rios con muchos peligros. Este juego ha encontrado su lugar en la cultura popular, incluyendo la televisión y la música. El objetivo del programa realizado es desarrollar una aplicación utilizando el lenguaje de programación Ensamblador, que permita comprender el arranque de un Sistema Operativo.

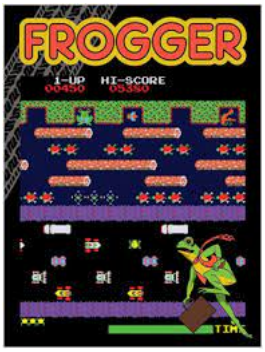

Figure 1: Imagen del juego Frogger.

En nuestro programa, el juego Frogger consiste en una rana que trata de pasar por almenos 3 calles, en las cuales transitan diferentes tipos de carros. Los automoviles tienen un largo de 1, las busetas tienen un largo de 2 y los camiones corrsponden a largo 3. Si se logra pasar por todos los carriles sin que sea colisionado, el jugador gana.

1.1 Requerimientos

- Se debe utilizar el lenguaje de programación Ensamblador para x86.
- EFI será el mecanismo de booteo.

- Programar en ensamblador el booteo desde una unidad de USB.
- Una vez que bootee desde el USB, cargará única y exclusivamente un programa llamado: Frogger.

2 Ambiente de desarrollo

Las herramientas utilizadas para la elaboración y la implemnetación de la tarea corta 2 fueron las siguientes:

- Se utilizó una laptop HP, con 12 GB de RAM y un sistema x64.
- Un Sistema Operativo Ubuntu 20.04, el cual fue trabajado sobre una Oracle VirtualBox.
- Las pruebas fueron realizadas sobre la VirtualBox.
- El compilador del código ensamblador x86 fue Fasm, aunque también se realizaron pruebas en Masm y Nasm.
- Se utiliza la biblioteca uefi.inc para importar los diferentes headers necesarios en el código.

3 Estructuras de datos usadas y Funciones

3.1 Estructuras

La estructura utilizada es una matriz, en la cual existen 5 carriles por donde pasan los carros. Los carros pasan por los carriles del medio y las filas 1 y 5 son los carriles seguros. Si el jugador llega a la fila 1 significa que ya ganó.

3.2 Funciones

Entre las funciones más importantes se encuentran:

- Mover el frog: Estas funciones toman la entrada del usuario y realizan el movimiento del frog en el tablero dependiendo de la entrada.
- Mover los vehiculos: Esta función realiza el movimiento de los vehículos sobre los carriles del juego, además implementa funciones de validación para no salirse del tamaño de la matriz.
- Funciones de validación: Validan el movimiento de todos los agentes presentes en el juego, asegurandose de que no se salgan del tablero si llegan a un punto donde ya no se puede mover. También ayudan a agilizar el movimiento de los vehículos y el frog.

- Funciones de finalización: Estas funciones validan si el frog se encuentra en la primera posición de la matriz, lo cual lo convierte en un ganador, o por el contrario choca con algún vehículo y pierde.
- Funciones de la biblioteca uefi.inc, las cuales nos permiten realizar operaciones sobre pantalla, imprimir datos, limpiar la pantalla y obtener los datos desde el teclado.

4 Instrucciones para ejecutar el programa

Los pasos son los siguientes:

1. Primeramente se debe instalar Fasm mediante el comando - sudo apt-get install -y fasm.
2. Luego creamos una carpeta donde se va a encontrar el proyecto. La carpeta puede tener cualquier nombre.
3. Dentro de esta carpeta se crearán dos carpetas más. Una carpeta llamada efi y dentro de esta carpeta se creará otra llamada boot.
4. Ya con las carpetas creadas, abrimos una terminal en la cual vamos a correr el siguiente comando - fasm frogger.asm.
5. Al ejecutar el comando, se creará un archivo frogger.efi, este archivo debe estar dentro de la carpeta boot y se le debe cambiar el nombre a "bootx64.efi".
6. Ya con el archivo "bootx64.efi" dentro de boot, ejecutamos el siguiente comando - mkisofs -o frogger.iso ./carpeta, donde carpeta es el nombre de la carpeta que creamos al inicio.
7. Este comando nos creará el Frogger.iso, que lo utilizaremos proximamente en la máquina virtual.
8. Creamos una nueva máquina virtual con las opciones por defecto, en las cual iremos a su configuración y en el apartado de sistema habilitaremos el Efi. Además de esto, debemos agregar el Frogger.iso en la parte de almacenamiento.
9. Por último, inicializamos la máquina virtual y jugamos.

5 Autoevaluación

El estado final del Frogger es que cumple con la mayoría de los requerimientos solicitados, aunque al principio se tuvieron algunos problemas para correr el código ya que presentaba algunos errores que no entendía. Además, trabajar sobre la máquina virtual hace que el trabajo se vuelva más lento, ya que no

tiene tantos recursos y se vuelve lenta e incluso en algunos momentos se quedaba pegada. La calificación incluida con la rúbrica de "Evaluación" es:

- Sector de Arranque: 30
- Frogger: 45
- Documentación: 17

El link del repositorio es https://github.com/andres0599/Tarea2-Frogger.git. No sabía que tenía que subir al git la información entonces hasta ahora la subo.

6 Actividades realizadas por el estudiante

Lista de Actividades		
Actividad	Día	Cantidad Horas
Investigar sobre ensamblador	22/3/2020	3 hrs
Instalar la máquina virtual	23/3/2020	2 hrs
Instalar los diferentes ambientes de trabajo	24/3/2020	3 hrs
Códificar y probar ambiente	24/3/2020	2 hrs
Investigar sobre booteo con EFI	26/3/2020	4 hrs
Programar funciones básicas del Frogger	28/3/2020	4 hrs
Programar movilidad de los autos	1/4/2020	2 hrs
Programar movilidad del Frog	3/4/2020	4 hrs
Empezar documentación	4/4/2020	2 hrs
Documentar el código	4/4/2020	4 hrs
Armar el loop del juego	5/4/2020	3 hrs
Realizar el booteo	6/4/2020	7 hrs
Realizar pruebas	6/4/2020	4 hrs
Terminar documentación	29/3/2020	3 hrs

7 Lecciones Aprendidas

La mayor lección que aprendí fue dedicar la mayor parte de tiempo a investigar sobre codificar, ya que es mucho más fácil codificar algo que usted entiende. En lenguajes como ensamblador o rust no hay mucha documnetación o videos que ayuden a entender las cosas, así que si no encuentra información hay que seguir buscando y tratar de entender y analizar todo para poder cumplir los objetivos. Ensamblador es un lenguaje difícil, pero también es muy repetitivo, por lo cual el código de las funciones se repite mucho y con la práctica se logra entender y mejorar rápidamente. El booteo fue los que menos me costó, ya que tomo menos tiempo, pero no es recomendable dejar todo para el final. Distribuir las horas de trabajo y distribuir bien las actividades por realizar es de suma importancia para lograr todos los objetivos y no tener un desgaste mental y físico muy severo.

References

[1] Kevin Moraga.(2021).Documentación Tarea 2

[2] Wikipedia contributors. (2021, 27 marzo). Frogger. Wikipedia. https://en.wikipedia.org/wiki/Frogger

[3] EFIBootLoaders - Ubuntu Wiki. (s. f.). Ubuntu Wiki. https://wiki.ubuntu.com/EFIBootLoaders

[4] Guide to x86 Assembly. (s. f.). Guide. https://www.cs.virginia.edu/

[5] Uefi. (s. f.). Wiki. https://wiki.osdev.org/Uefi.inc