Videojuego en ensamblador x86_64 para sistemas operativos Linux

Johan Arrieta Solórzano
johnarso@gmail.com
Jafet Chaves Barrantes
jafet.a15@gmail.com
Melissa Fonseca Rodríguez
mfonsecarodriguez83@gmail.com
Dayhana Sánchez Jiménez
dayhana.sj@gmail.com

***Resumen*—En este informe se presenta el desarrollo de un video juego similar al Arkanoid, el cual fue programado en el lenguaje ensamblador x86_64, utilizando NASM, por lo que se realiza la programación en su más bajo nivel, y utilizando el sistema operativo de la familia Linux, cumpliendo los requerimientos establecidos.**

***Index Terms*—Ensamblador x86, registros, videojuego, NASM**

I. Introducción

El videojuego Arkanoid es un juego clásico de la década de los 80's, el cual consiste en mover una plataforma de manera que se puedan eliminar todos los bloques de la parte superior, e impidiendo que la bola caiga de la plataforma, por la parte inferior, por lo que este proyecto consistió en la programación de este juego, utilizando el lenguaje ensamblador en la plataforma de Linux 64 bits, con el objetivo de desarrollar habilidades prácticas y conocimiento en el manejo de los recursos de hardware, y programación en bajo nivel.

Realizar la programación del juego fue un reto para el grupo, debido a que la programación en bajo nivel es más complicada que la programación en alto nivel y se está acostumbrado a trabajar en alto nivel; sin embargo, se logra cumplir con las expectativas, logrando crear un juego funcional para ejecutarlo en la terminal del sistema operativo Linux. En este documento se presenta el problema, la solución planteada y las estrategias utilizadas para llegar a cumplir con dicho objetivo.

II. Marco Teórico

II-A. Arquitectura x86-64

Esta arquitectura es la versión de 64 bits del conjunto de instrucciones x86. El conjunto de instrucciones x86-64 fue diseñado por AMD y ha sido implementada por Intel, VIA y otros. La tecnología 86_64 está diseñada para permitir a los proveedores de la plataforma, los desarrolladores y las empresas la transición de entornos de 32 bits a 64 bits sin dejar de tener un rendimiento líder en la gran base instalada de aplicaciones de 32 bits existentes. La tecnología de 64 bits es ideal para aplicaciones que consumen mucha memoria, como las grandes bases de datos, herramientas de CAD, y los motores de simulación que actualmente están limitados por la escasez de memoria que representan 4 GB. (Internet Archive Wayback Machine, 2012).

Dentro de sus ventajas están:

- Soporta una cantidad mucho mayor de memoria virtual y memoria física, permitiendo almacenar mayor cantidad de datos.
- Provee registros de uso general de 64 bits.
- Los compiladores producen binarios que funcionan tanto en AMD64 como Intel64.
- Puede ejecutar aplicaciones para 32 bits.
- Incluye un soporte nativo para ejecutar aplicaciones de 16 bits.

Sus desventajas son:

- Existen diferencias con el conjunto de instrucciones de AMD64 e Intel 64.

En cuanto a la utilización de esta arquitectura en el sistema operativo Linux, se puede aprovechar las características de este sistema, ya que Linux proporciona compatibilidad hacia atrás para el funcionamiento de los ejecutables de 32 bits, además Linux 64 permite hasta 128 TB de espacio en direcciones virtuales para los procesos individuales, y puede hacer frente aproximadamente a 64 TB de memoria física, sujeto a las limitaciones del procesador y del sistema.

II-B. Ambiente de trabajo

La programación del juego se realizó en el lenguaje ensamblador, en el sistema operativo de Linux en las distribuciones de Lubuntu y Ubuntu, por lo que antes de iniciar este proyecto fue necesario la virtualización de estos sistemas operativos por medio de una máquina virtual.

Los pasos utilizados en esta primera etapa fueron:

- Instalar Virtual Box
- Construir la máquina virtual
- Descargar la versión de Linux (Se recomienda Lubuntu ya que es una versión más liviana).

- Instalar el ensamblador Netwide Assembler (NASM), el cual convierte un archivo a código objeto o a un archivo código máquina.
- Instalar algún editor de texto, en el presente caso Sublime Text.
- Iniciar a escribir el código.

Debido a que el sistema nativo de las PC's es Windows, se instala una máquina virtual para correr el sistema operativo Linux. https://www.virtualbox.org

La distribución de Linux a utilizar es Lubuntu, por ser una versión de Linux más liviana. http://lubuntu.net/

Para la instalación de programas desde Linux (Lubuntu) se utilizan los comandos "apt-get install ...", y el programa a instalar en la terminal. De esta manera se instala el debugger el cual permite ejecutar el programa de manera controlada y revisar el movimiento de los datos en los registros y la memoria del computador, para instalar este paquete se escribe en la terminal "gdb".

II-C. Otras consideraciones

Se debe comprender que el microprocesador cuenta con registros internos, los cuales se utilizan para almacenar valores constantes o para datos procesados a partir de las instrucciones. Para la arquitectura x86_64 se cuenta con la siguiente lista de registros:

Tabla I
REGISTROS PARA LA ARQUITECTURA X86-64

Número	Nombre	Propósito
0	rax	Registro acumulador
1	rbx	Registro base
2	rcx	Registro contador 4to Arg
3	rdx	3er Argumento de llamada al sistema
4	rsi	2do Argumento de llamada al sistema
5	rdi	1er Argumento de llamada al sistema
6	rbp	Base pointer
7	rsp	Stack Pointer
8-15	r8 - r15	Propósito variado no específico

III. SOLUCIÓN IMPLEMENTADA

Para la realización del proyecto, se decidió dividir el desarrollo de la programación en varias etapas, esto según las tres pantallas a mostrar en el juego.

III-A. Pantalla de inicio

Según los requerimientos del proyecto, en la pantalla de inicio se deben mostrar:

- Mensaje de bienvenida
- Solicitar nombre del jugador
- Mostrar nombre del curso y semestre

El código para mostrar un mensaje en pantalla lleva la estructura de la Fig. 1

```
Section .data
        cons_bienvenida: db 'Bienvenido a Micronoid', 0xa
        Cons_tamano: equ $-cons_bienvenida
Section .text
        Global _start
_start:
        mov rax,1
        mov rdi,1
        mov rsi,cons_bienvenida
        mov rdx,cons_tamano
        syscall
```

Figura 1. Sección de código para imprimir un mensaje en pantalla

Debido a que tanto para el mensaje de bienvenida como en la pantalla de juego y en la pantalla final se debe usar el mismo código para imprimir un mensaje, se utilizan macros que es una estructura de sintáxis que permite al lenguaje ensamblador reutilizar una sección de código más de una vez, y reducir así el tamaño del código total, ésta es una opción análoga a las funciones en lenguajes de más alto nivel.

La estructura utilizada para macros se presenta en la Fig. 2

```
%macro limpiar_pantalla 2       ;recibe 2 parametros
        mov rax,1               ;sys_write
        mov rdi,1               ;std_out
        mov rsi,%1              ;primer parametro: caracteres especiales para
limpiar la pantalla
        mov rdx,%2              ;segundo parametro: Tamaño
        syscall
%endmacro;------------------------------------------------------------------
```

Figura 2. Sección de código para imprimir un mensaje utilizando macros

III-B. Pantalla de juego

La pantalla de juego es en la cual el usuario puede interactuar con los bloques y con la plataforma, por lo que esta pantalla debe cumplir con ciertos requerimientos, entre estos:

- Mostrar tres filas de seis bloques en la parte superior de la pantalla.
- Mostrar una plataforma en la parte inferior, la cual debe ser capaz de moverse de izquierda a derecha.
- Mostrar una bola al inicio del juego centrada en la plataforma la cual debe ser capaz de rebotar en los bloques y paredes.
- Desaparecer los bloques al momento que la bola choque con ellos.
- Mostrar el nombre del jugador y la cantidad de vidas disponibles.
- Indicar al jugador presionar una tecla para iniciar el juego.
- Mostrar un mensaje al usuario en caso de que pierda una vida.
- Mostrar un mensaje al usuario en caso de que gane el juego.
- Mostrar un mensaje al usuario en caso de que pierda el juego por completo.

III-B1. Filas de bloques: Para realizar los bloques del juego, así como el recuadro que enmarca toda la interfaz, se utilizan las secuencias de escape ANSI, con las cuales se trabajan los bloques como caracteres, siendo posible así usar diferentes características para cada uno.

Las dimensiones utilizadas para el marco es de 110 caracteres por 42, por lo que se realizan los bloques, de manera que estos cubran por completo el ancho de juego. El ancho utilizado para cada bloque es de 18 espacios.

Cada uno de los bloques es programado de manera independiente, de manera que al momento en que la bola choque con alguno, se facilite que estos desaparezcan tanto visualmente como en el espacio, de forma que si la bola vuelva a pasar por el mismo lugar en donde ya ha desaparecido un bloque no rebote con este bloque fantasma.

Para simular la desaparición de alguno de los bloques, se procede a imprimir un espacio en blanco del tamaño del bloque, de manera que este deje de ser visible y no afecte el recorrido de la bola.

El código utilizado para definir los bloques se presenta en la Fig. 3, el cual es el mismo para las tres filas de bloques, a diferencia de los colores utilizados

```
b11: db  0x1b, "[5;4f",0x1b, "[42;32m",  '                  ', 0x1b, "[40;37m"
b_size: equ $-b11
b12: db 0x1b, "[5;22f", 0x1b, "[45;35m", '                  ', 0x1b, "[40;37m"
b13: db 0x1b, "[5;40f", 0x1b, "[46;36m", '                  ', 0x1b, "[40;37m"
b14: db 0x1b, "[5;58f", 0x1b, "[44;34m", '                  ', 0x1b, "[40;37m"
b15: db 0x1b, "[5;76f", 0x1b, "[41;31m", '                  ', 0x1b, "[40;37m"
b16: db 0x1b, "[5;94f", 0x1b, "[43;33m", '                  ', 0x1b, "[40;37m"
```

Figura 3. Sección de código utilizado para imprimir los bloques de la primera fila

En la Fig. 4 se especifican los colores utilizados en las secuencias de escape

Intensity	0	1	2	3	4	5	6	7
Normal	Black	Red	Green	Yellow[16]	Blue	Magenta	Cyan	White
Bright	Black	Red	Green	Yellow	Blue	Magenta	Cyan	White

Figura 4. Colores utilizados para texto y fondo

III-B2. Plataforma: La plataforma al igual que los bloques se realiza con caracteres y secuencias de escape para dar color a la barra, el propósito de la barra es que se mueva de izquierda a derecha sobre la misma fila, de manera que no se salga de los bordes establecidos por el marco de la interfaz, de igual forma la plataforma no se puede hacer mas pequeña o esconder parte de la barra cuando esta se acerca al borde.

Para hacer posible el desplazamiento de la plataforma, ésta está en constante refrescamiento, es decir, cada cierto tiempo está volviéndose a escribir y cuando se presiona una tecla de las establecidas para el desplazamiento, se borra la plataforma escribiendo espacios a lo largo de la fila en donde está posicionada y se vuelve a colocar con el aumento en la posición horizontal correspondiente a la tecla presionada.

A manera de evitar que la plataforma se pase de los límites establecidos en la interfaz cada vez que se va a realizar el aumento en la dirección derecha o izquierda se realiza una comparación del registro correspondiente con tanto el límite derecho como izquierdo y si se da una igualdad entonces no se genera el desplazamiento.

III-B3. Bola: La bola es la figura u objeto que se mueve a lo largo de toda la interfaz definido dentro del marco del juego, la cual al momento de colisionar con alguno de los bloques rebota hacia la parte inferior, ya sea hacia las paredes o directamente hacia la plataforma.

La bola debe cumplir con los siguientes requerimientos:

- Al inicio del juego debe aparecer sobre y centrada la plataforma.
- La bola no debe moverse hasta el momento de presionar la tecla X.
- Salir de la plataforma con un ángulo entre $5°$ y $85°$.
- El ángulo de rebote debe ser de $90°$.
- La bola no puede evadir el espacio dejado por los bloques que han sido desaparecidos.
- La bola debe mantener una velocidad constante durante todo el recorrido.

III-C. Pantalla final

La pantalla final aparece una vez el usuario ha ganado o perdido el juego por completo, la lógica empleada es similar a la implementación de la pantalla inicial con la diferencia en la información que se muestra. En ella se colocan los mensajes:

- Mostrar un mensaje al final ”Gracias por jugar Micronoid”
- Mostrar los nombres de los integrantes del grupo.
- Mostrar el tipo de procesador y la familia a la que pertenece.

IV. Resultados

La presentación de la pantalla de bienvenida de juego se presenta en la Fig. 5, la cual es la primera que ve el usuario al momento de correr el ejecutable, y en la cual se le solicita ingresar el nombre.

La presentación de la interfaz de juego se presenta en la Fig. 6, en la cual el usuario arranca el juego y puede comenzar a eliminar bloques después de haber presionado la tecla x, hasta lograr eliminar todos los bloques o bien perder una vida (caer al piso).

En la Fig. 7 se presenta el mensaje final en caso de que el usuario pierda el juego.

Figura 5. Pantalla de bienvenida del juego

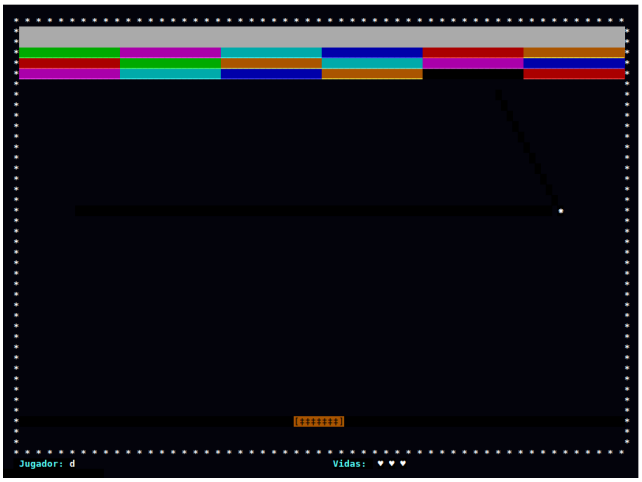

Figura 6. Pantalla de interfaz del juego

Figura 7. Pantalla del fin del juego en caso que el usuario pierda

En la Fig.8 se presenta la pantalla final en caso de que el usuario gane el juego.

Figura 8. Pantalla del fin del juego en caso que el usuario gane

La Fig.9 ilustra esta última sección en la ejecución del videojuego.

Figura 9. Pantalla de despedida del juego

Como se logra ver en la Fig. 5, Fig. 6, Fig. 7, Fig. 8, Fig. 9, gráficamente se logra apreciar que el juego cumple con los requerimientos, pues se muestra el nombre del jugador y la cantidad de vidas en la parte inferior del juego, además de los otros requerimientos anteriormente enumerados. Respecto a la parte funcionalidad del juego, se puede decir que se cumplieron con todas las especificaciones pero algunas de ellas tienen algunos problemas dependiendo de como se utilice el juego, estas fallas se mencionan en la subsección IV-A1.

IV-A. Limitaciones y Recomendaciones

IV-A1. Limitaciones:

- El juego puede ser ejecutado en un ordenador que utilice el sistema operativo Linux.
- EL procesador del ordenador debe ser específicamente de una arquitectura x86-64 bits.

- El juego no cuenta con el desarrollo de colisiones laterales.
- Si se mantiene presionada una tecla que no sea una de las establecidas la bola se mueve con mayor rapidez, lo mismo sucede al mantener presionada una tecla de las establecidas pero en menor medida.
- Si se presiona x para iniciar el juego y no se presiona ninguna otra tecla al perder una vida se vuelve a inciar el juego, esto debido a que en el buffer se mantiene la tecla x y cuando se vuelve a leer se va a observar la x.

IV-A2. Recomendaciones:

- Analizar el código de manera que sea posible reducir la cantidad de código que el procesador tiene que ejecutar.
- Buscar la manera de hacer compatible el juego con sistemas operativos de 32 bits, de y familias como Windows y Mac.
- Implementar mayor cantidad de bloques, de niveles, un menú, y un sistema de puntuación en el cual la complejidad del juego aumente.
- Para involucrar más al usuario en el juego puede ser posible incorporarle música al video juego.
- Al desarrollar el juego tener un buen manejo de los registros, para evitar que sucedan errores, ya que al ser pocos registros y utilizarlos en varios lugares de la programación puede generar errores por el cambio de valores en los mismos que no estaban contemplados.
- Utilizar el stack para almacenar valores momentaneos, pero tener cuidado al momento de hacer los pops correspondientes, pues se pueden intercambiar valores o bien hacer pop de direcciones necesarias para volver a rutinas.

V. Conclusiones

- La programación en ensamblador es más compleja al ser en bajo nivel y requiere mantener un mejor control sobre el flujo de datos en el sistema, sin embargo se logró terminar el juego de manera funcional cumpliendo con los requerimientos establecidos.
- El uso de la programación en bajo nivel permitió conocer con mayor detalle como el procesador ejecuta las instrucciones, así como el manejo de la pila para el almacenamiento de variables y registros.
- Utilizar sistemas operativos de la familia GNU/Linux permitió demostrar su potencial como sistema computacional de programación.

Referencias

[1] Guía de inicio. Lenguaje ensamblador x86_64 para Linux. *Parte 1: Ensamblado del primer programa*. Escuela Ingeniería Electrónica.

[2] NEOMATRIX. Curso ensamblador x86. *M*anipulando la Pantalla. [online]. (27 Junio 2015) Disponible en: https://www.youtube.com/watch?v=CeKZo8HvEUo

[3] Orenga, M; Manonellas, G.(s.f) *Programación en ensamblador x86-64*. Universitat Oberta de Catalunya.

[4] Seyfarth, R.(2011) *An Introduction to 64 bit Intel Assembly Language Programming for Linux*. Hattiesburg, MS, USA: School of Computing, University of Southern Mississippi.

[5] Wikipedia. *ANSI escape code*. [online] (16 Julio 2016). Disponible en: https://en.wikipedia.org/wiki/ANSI_escape_code