Tarea 1: Mobile maze booteable

Mario Araya Chacón
2018319178
Tecnológico de Costa Rica
.

Fabián Ramírez Arrieta
2018099536
Tecnológico de Costa Rica
.

Tomás Segura Monge
2018099729
Tecnológico de Costa Rica
.

Wajib Zaglul Chinchilla
2018099304
Tecnológico de Costa Rica
.

***Resumen*—Este proyecto se enfocó en desarrollar un juego de laberinto en lenguaje ensamblador x86 con un bootloader personalizado. Se implementó la renderización en pantalla a través de interrupciones de software y se adquirió conocimiento sobre la estructura del sistema de arranque de un sistema operativo. El juego es funcional y se logró una comprensión profunda sobre la programación de bajo nivel y sistemas operativos. Se recomienda optimizar el código para mejorar el rendimiento y explorar otras técnicas de renderización. En general, este proyecto fue una experiencia de aprendizaje valiosa en programación de bajo nivel.**

***Index Terms*—x86, interrupciones de sistema, juego, ensamblador, bootloader.**

I. Introducción

I-1. Arquitectura x8086: La arquitectura x86 es una arquitectura de procesadores que se utiliza en la mayoría de los sistemas informáticos personales. El procesador x86 original fue lanzado en 1978 por Intel, y su sucesor, el x8086, fue lanzado en 1979. La arquitectura x8086 es la base para la mayoría de los procesadores Intel de la actualidad. Esta arquitectura utiliza un conjunto de instrucciones complejas (CISC). La mayoría de los sistemas operativos modernos se han diseñado para funcionar en procesadores x86 [1].

I-2. QEMU: QEMU es un programa de emulación de sistema que permite ejecutar sistemas operativos y aplicaciones diseñados para diferentes arquitecturas de procesadores en un sistema anfitrión. QEMU es un software libre y gratuito que se puede utilizar en una amplia variedad de sistemas operativos, incluyendo Windows, macOS, Linux y otros [2].

QEMU utiliza una técnica llamada virtualización para crear un entorno de ejecución aislado para el sistema invitado. Esto permite a los usuarios ejecutar sistemas operativos y aplicaciones en diferentes arquitecturas de procesadores sin necesidad de hardware adicional [2].

I-3. Boot loader: El boot loader o cargador de arranque es un programa que se ejecuta al inicio del sistema operativo para cargar el kernel en la memoria del sistema. El boot loader se encarga de realizar una serie de comprobaciones y configuraciones del hardware antes de cargar el kernel [3].

El boot loader también proporciona una interfaz para que el usuario pueda seleccionar el sistema operativo que desea cargar si hay varios sistemas operativos instalados en el ordenador [3].

Identify applicable funding agency here. If none, delete this.

I-4. Interrupciones del sistema operativo: Las interrupciones del sistema operativo en x8086 son eventos que detienen temporalmente la ejecución de un programa y transfieren el control al kernel del sistema operativo. El kernel se encarga de gestionar la interrupción y proporcionar el servicio correspondiente, como la lectura de un archivo o el envío de datos a través de una red. Estas interrupciones se identifican mediante un número, conocido como "vector de interrupción", y se pueden invocar desde el código de usuario mediante una instrucción especial. La tabla de vectores de interrupción se encuentra en la memoria del sistema y contiene las direcciones de las rutinas de servicio correspondientes a cada número de interrupción [4].

A lo largo de este documento se presentará el proceso para el desarrollo de una bootloader encargada de cargar un videojuego a partir de un usb booteable. Se presentará el ambiente de desarrollo, detalles de diseño y características administrativas del proceso de desarrollo.

II. Ambiente de desarrollo

Para poder desarrollar de manera efectiva el proyecto se utilizó las siguientes herramientas.

II-A. CMake

Un archivo Makefile contiene las recetas de comandos en consola que permiten compilar y ejecutar el proyecto.

Instalación de CMake: *sudo apt-get -y install cmake.*

II-B. Linux

El programa se desarrolló sobre el sistema operativo Linux y la distribución Ubuntu. También se probó haciendo uso de Windows Subsystem for Linux (WSL) 2. Para descargar e instalar WSL2 se recomienda seguir los pasos documentados a continuación: Install WSL | Microsoft Learn. Adicionalmente, se deben instalar los drivers necesarios para ejecutar interfaces gráficas como se explica en Run Linux GUI apps with WSL | Microsoft Learn.

En caso de querer descargar e instalar el sistema operativo completo, se recomienda la siguiente guía de instalación: Install Ubuntu desktop | Ubuntu.

II-C. Qemu

Qemu permite correr el juego en su forma gráfica y una guía de instalación es la siguiente How To Install And Configure QEMU In Ubuntu | Unixmen.

III. Atributos

III-A. Aprendizaje continuo

El conocimiento adquirido durante el desarrollo del proyecto "Mobile Maze booteable"ha sido invaluable. En primer lugar, el proyecto ha requerido un conocimiento detallado de los fundamentos de la programación en lenguaje ensamblador x86, así como la implementación de rutinas de entrada/salida para manejar la pantalla y el teclado del sistema. Además, la implementación de un cargador de arranque ha requerido una comprensión del formato de los archivos de imagen de disco y la estructura del sector de arranque.

La estrategia implementada para satisfacer las necesidades de aprendizaje ha sido una combinación de investigación y práctica. En primer lugar, se ha realizado una exhaustiva investigación sobre las especificaciones del lenguaje ensamblador x86, así como sobre el formato de los archivos de imagen de disco y el sector de arranque. Además, se ha consultado una variedad de fuentes, como manuales de programación, foros y tutoriales en línea, para obtener una comprensión más profunda de los conceptos y técnicas necesarios para desarrollar el proyecto. Por otro lado, se ha puesto en práctica todo lo aprendido a través de la implementación de rutinas de entrada/salida, la creación de un cargador de arranque y la implementación de la lógica del juego en sí. En resumen, la combinación de investigación y práctica ha sido esencial para adquirir el conocimiento y las habilidades necesarias para desarrollar el proyecto de manera efectiva.

III-B. Trabajo individual y en equipo

III-B1. Estrategias de trabajo: Con el fin de organizar el trabajo individual y en equipo de manera equitativa e inclusiva a lo largo de las etapas del proyecto (planificación, ejecución y evaluación), se definieron las siguientes estrategias:

- Tomando como base que la mayoría de los miembros del equipo trabajan durante la mañana, se realizaron las reuniones de equipo durante las tardes y las noches.
- Se dividió la investigación sobre herramientas en 4 partes equitativas, una para cada miembro: bootloader, utilización de qemu, interrupciones en x86 y programación en ensamblador.
- A la hora de ejecución: se definieron reuniones por las noches durante las semanas para programar en conjunto con el fin de evitar problemas de integración. Los cuatro miembros del equipo se conectaron a cada una de las sesiones.
- Cuando se finalizó: en una sesión grupal se probó y evaluó el programa para verificar que cumple con los requerimientos.

III-B2. Planificación del equipo: Para la correcta planificación del trabajo fue necesario distribuir los miembros según roles (Table I), definir metas (Table II) y reglas de comportamiento y ejecución.

III-B2a. Reglas para el comportamiento y ejecución del proyecto:

- Las reuniones de equipo serán definidas según la disponibilidad de horario de cada miembro y todos deben aceptar la fecha y hora de la reunión.
- La repartición de roles es flexible, lo que quiere decir que un rol no es exclusivo de los miembros asignados. Cualquier miembro puede asumir tareas de otro rol que no se le fue asignado. Esto siempre y cuando beneficie el rendimiento y ambiente de trabajo.
- Se utilizará un repositorio en Git donde se hará push cada vez que se tenga una nueva funcionalidad ejecutándose de manera correcta.
- Comunicación abierta y efectiva: asegurarse de que todos los miembros del equipo se comuniquen de manera clara y respetuosa. Esto implica escuchar atentamente, hacer preguntas pertinentes, ser honesto y expresarse de forma clara y directa.
- Responsabilidad individual: cada miembro debe estar dispuesto a cumplir con sus responsabilidades y compromisos, y a informar de cualquier problema o retraso que pueda afectar al trabajo del equipo.
- Trabajo en equipo: colaborar y trabajar juntos para lograr los objetivos del equipo. Esto implica respetar las diferencias individuales y valorar las habilidades y experiencias de cada miembro.

III-B3. Acciones asertivas: A continuación se definen algunas de las acciones que promueven la colaboración entre los miembros del equipo durante el desarrollo del proyecto:

- Definir objetivos claros: permite establecer un plan de trabajo claro y efectivo.
- Establecer un plan de trabajo. Establecer un plan de trabajo detallado, que incluya los pasos a seguir, los plazos y las responsabilidades de cada miembro del equipo de modo que todos los miembros se identifiquen con él.
- Comunicación efectiva: Mantener una comunicación efectiva y constante con todos los miembros del equipo, para asegurarse de que todos estén alineados y trabajen juntos hacia los mismos objetivos.
- Celebrar los logros en equipo: Al celebrar los logros y éxitos del equipo, se fomenta la motivación, el compromiso y la cohesión del equipo.
- Resolver conflictos de manera constructiva: los conflictos se abordan de manera constructiva, escuchando todas las perspectivas y buscando soluciones que beneficien a todos.
- Considerar e incluir las diferentes formas de pensar. Así como el respeto hacia creencias religiosas, orientaciones sexuales, discapacidades y demás aspectos de diversidad.

III-B4. Ejecución de estrategias: Las estrategias planificadas para el proyecto deben ser ejecutadas de manera correcta, por lo que se definen las siguientes recomendaciones:

- Apegarse a la calendarización y las metas definidas.
- Cada miembro debe ejecutar su rol con diligencia y sensatez.

Cuadro I
DISTRIBUCIÓN DE LOS MIEMBROS DEL EQUIPO SEGÚN ROLES Y SUS FUNCIONES.

Rol	**Funciones**	**Miembros**
Programador	Implementar y modificar estrategias y lógica computacional en ensamblador x86.	Fabián, Wajib, Tomás, Mario.
Diseñador de software.	Investigar y diseñar estrategias previas para la implementación del programa a nivel de software.	Fabián, Mario.
Líder de organización.	Verifica que el equipo esté al día con sus tareas y concluya los objetivos según las metas.	Tomás.

Cuadro II
DEFINICIÓN DE LAS METAS DEL PROYECTO, REQUERIMIENTOS NECESARIOS Y FECHA DE ENTREGA ASOCIADA.

Meta	**Requerimientos**	**Fecha de entrega aproximada**
Bootloader 0.2	Investigar, planificar y diseñar los requerimientos del programa.	2 de Marzo
Bootloader 0.7	Implementación del bootloader y renderización del menú principal del juego. Uso de interrupciones gráficas.	9 de Marzo.
Bootloader 1.0	Implementación total de los requerimientos del sistema. Evaluación del programa en base a los requerimientos. Completitud de la documentación solicitada.	14 de Marzo.

- En caso de encontrar problemas y no poder resolverlas, el miembro debe comunicar estos problemas al resto del equipo para juntos decidir cómo se procede.
- Comunicación asertiva a lo largo de todo el proceso.
- El orden para ejecución de las estrategias debe ser: definición de objetivos, ejecución de las metas definidas y evaluación del proyecto y el trabajo en equipo.

III-B5. Evaluación para el trabajo en equipo e individual:

III-B5a. Evaluación del desempeño individual: Cada miembro del equipo fue asignado a un rol. Se verifica que cada miembro cumpla de manera efectiva con las funciones asignadas de su rol.

Cada uno de los miembros debe estar de acuerdo con el correcto cumplimientos de las tareas individuales de los demás miembros. Además, debe coincidir en que cada miembro cumplió con las acciones asertivas y reglas de comportamiento definidas anteriormente.

Finalmente, cada miembro tiene derecho a contrastar la evaluación de los demás miembros contra una autoevaluación y conserva el derecho al desacuerdo y el debate.

III-B5b. Evaluación del desempeño en equipo: En la especificación del proyecto se definieron los requerimientos del programa. En base a ellos se verifica el cumplimiento de cada una de las características especificadas. Esto permite evaluar el desempeño técnico.

Se define una reunión final en donde se debate sobre el cumplimiento de las estrategias para la ejecución del proyecto. Cada miembro debe dar su opinión y cómo se ha sentido respecto a:

- Si la distribución de trabajo fue justa.
- Si el ambiente de trabajo fue sano.
- Si el cumplimiento de los objetivos satisfizo las expectativas.
- Si el equipo tuvo cohesión para ejecutar las tareas.

III-B5c. Evaluación para las estrategias de equidad e inclusión: Durante la reunión de evaluación en equipo se plantean las siguientes preguntas adicionales:

- ¿Considera que la comunicación fue abierta, respetuosa y regular entre los miembros del equipo?
- ¿Considera que existió la discriminación de algún tipo durante la ejecución del proyecto?
- ¿Cree que se fomenta la participación y liderazgo inclusivo de cada miembro durante la ejecución del proyecto?
- ¿Cree que las tareas se distribuyeron de manera equitativa y justa entre los diferentes miembros?

III-B5d. Evaluación para las acciones asertivas: Durante la reunión para retrospectiva del equipo se plantean las siguientes interrogantes que se deben discutir entre los diferentes miembros de manera abierta e inclusiva:

- ¿Considera que los objetivos de trabajo se definieron de manera clara?
- ¿Considera que el plan de trabajo fue distribuido de manera justa?
- ¿Cree que el plan de trabajo establecía las pautas necesarias para facilitar el avance a lo largo del proyecto?
- ¿Cree que las metas propuestas eran realistas y alcanzables?
- ¿Considera que los logros se celebraron de la manera adecuada y satisficieron sus expectativas?
- ¿Cree que los conflictos se resolvieron de manera asertiva?
- ¿Cree que el equipo desempeñó sus tareas y mantuvo conductas inclusivas en el procesos?

IV. DETALLES DE DISEÑO

Se diseñaron tres principales diagramas:

- Diagrama de arquitectura (Fig. 1): muestra la relación entre el hardware de computadora, el usuario, las entradas y salidas del computador con el programa en sí.

- Diagrama de flujo para mostrar la relación entre las diferentes pantallas de juego, los principales ciclos de juego y el bootloader (Fig. 3).
- Diagrama de flujo para mostrar la relación de las interrupciones del sistema con los principales ciclos del juego (Fig. 3)

Además, se diseñaron las distribuciones de los obstáculos de cada nivel como se muestra en Fig. 5 y Fig. 5.

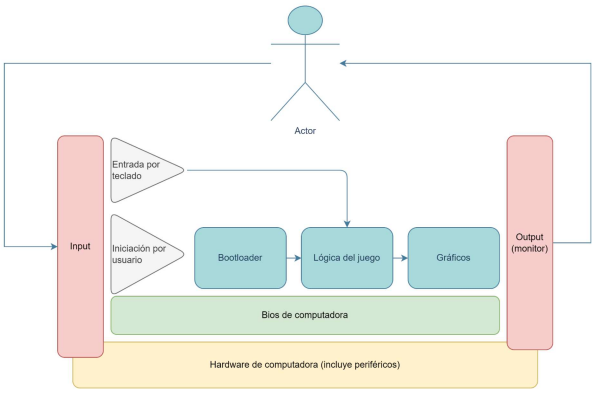

Figura 1. Diagrama de arquitectura del programa.

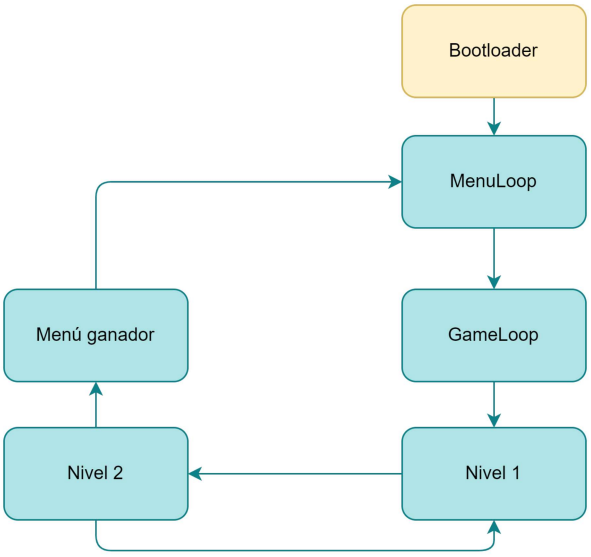

Figura 2. Diagrama de flujo para la relación entre las diferentes pantallas de juego, los principales ciclos de juego y el bootloader.

V. Instrucciones de uso

V-A. *Ejecución del programa*

Para ejecutar el proyecto es necesario instalar las librerías y programas mencionados en la sección de Ambiente de desarrollo. Posterior a su correcta implementación, se ejecuta el siguiente comando dentro de la raíz del directorio: *make*.

Este comando se encarga de ejecutar la receta para la compilación y el despliegue en pantalla de la interfaz gráfica.

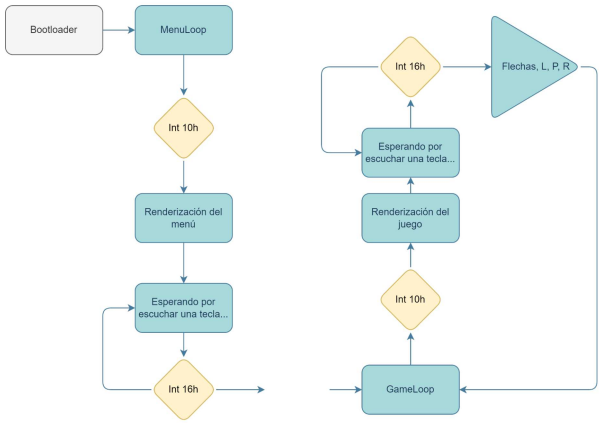

Figura 3. Diagrama de flujo para la relación de las interrupciones del sistema con los principales ciclos del juego.

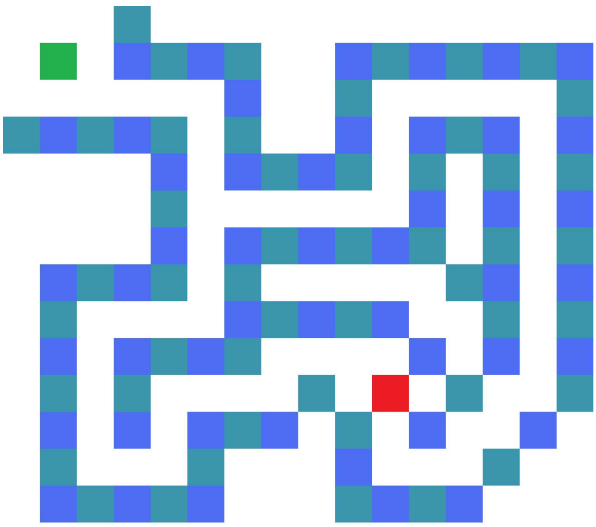

Figura 4. Diseño de mapa para nivel 2.

V-B. *Uso del teclado*

Posterior a la ejecución, las teclas para utilizar el juego son las siguientes:

- Flecha ↑: moverse hacia arriba.
- Flecha ↓: moverse hacia abajo.
- Flecha ←: moverse hacia la izquierda.
- Flecha →: moverse hacia la derecha.
- Espacio: permite iniciar el juego después de mostrar el menú.
- Tecla L: permite pausar el juego.
- Tecla R: reiniciar el juego.

V-C. *Reglas del juego*

El juego consiste en desplazar al jugador (cuadro verde) a través del tablero en dirección al cuadro rojo que es la meta. Al llegar al cuadro rojo cambia de nivel o gana el juego.

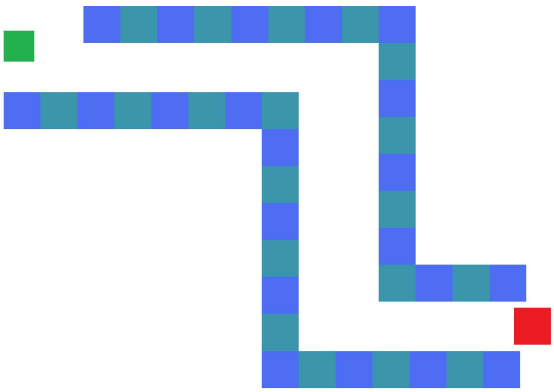

Figura 5. Diseño de mapa para nivel 1.

VI. Tabla de actividades

El resumen de las actividades se puede apreciar en las tablas Table VI, Table VI, Table VI y Table VI.

VII. Conlusión

Este proyecto ha implicado superar retos como implementar un bootloader funcional y mostrar gráficos en pantalla utilizando interrupciones. La implementación del bootloader ha sido un logro significativo, ya que ha permitido que el programa se cargue en la memoria del sistema y se ejecute sin problemas. Además, la implementación de interrupciones ha permitido la renderización de gráficos en la pantalla de manera eficiente y efectiva.

La implementación del bootloader ha requerido una comprensión profunda del formato de los archivos de imagen de disco y la estructura del sector de arranque. El código de arranque ha sido diseñado para cargar el programa en la memoria del sistema y pasar el control al punto de entrada en el código del juego. Además, la implementación de interrupciones ha permitido la manipulación de la pantalla de manera efectiva. El programa utiliza la interrupción de la BIOS INT 10H para manipular los registros del controlador de video y mostrar gráficos en pantalla.

En resumen, la implementación del bootloader y la renderización en pantalla a partir de interrupciones son logros significativos que demuestran la capacidad del proyecto para manipular y controlar el hardware de la computadora. Estos logros han sido posibles gracias a la comprensión profunda del lenguaje ensamblador x86 y la práctica de técnicas avanzadas de programación de bajo nivel.

VIII. Recomendación

Optimizar el código. Aunque el proyecto actualmente funciona correctamente, siempre hay espacio para mejorar el rendimiento. Se podría optimizar el código del juego para que funcione de manera más eficiente y reducir la cantidad de ciclos de reloj necesarios para completar una tarea. La optimización del código es una habilidad valiosa en el desarrollo de software, y puede mejorar significativamente el rendimiento del programa.

Optimizar la renderización en pantalla. La renderización en pantalla en este proyecto se realiza mediante interrupciones de software. Aunque esta técnica funciona, puede resultar lenta y consumir una gran cantidad de recursos del sistema. Para mejorar la eficiencia y reducir la carga del sistema, se podría considerar la implementación de renderizado por hardware, utilizando la tarjeta gráfica o la placa de video, lo cual puede resultar más rápido y eficiente.

Referencias

[1] Techopedia. (n.d.) What is x86? - definition from techopedia. [Online]. Available: https://www.techopedia.com/definition/20084/x86

[2] Q. Org. (n.d.) Qemu - a generic and open source machine emulator and virtualizer. [Online]. Available: https://www.qemu.org/

[3] IONOS. (n.d.) Bootloader: What is it and how does it work? [Online]. Available: https://www.ionos.com/digitalguide/server/configuration/what-is-a-bootloader/

[4] I. Corporation, "https://www.ionos.com/digitalguide/server/configuration/what-is-a-bootloader/," Intel Corporation, Tech. Rep., 2019. [Online]. Available: https://www.intel.es/content/www/es/es/architecture-and-technology/64-ia-32-architectures-software-developer-vol-3a-part-1-manual.html

Cuadro III
TABLA DE ACTIVIDADES Y HORAS APROXIMADAS DE TRABAJO - TOMÁS SEGURA

Rango de fechas	Actividades	Horas aproximadas
24 de Febrero - 2 de Marzo	Reunión de planificación del proyecto con el equipo de desarrollo. Establecimiento de objetivos y plazos para el proyecto. Investigación de tecnologías y herramientas de desarrollo necesarias para el proyecto. Configuración del entorno de desarrollo.	6
3 de Marzo - 9 de Marzo	Depuración de la funcionalidad básica del sistema. Integración de bibliotecas y herramientas externas. Pruebas de funcionalidad y depuración de errores.	14
10 de Marzo - 13 de Marzo	Redacción de la documentación. Soporte en pruebas adicionales de funcionalidad. Corrección de errores menores. Preparación para la revisión y demostración del proyecto ante el cliente.	8

Cuadro IV
TABLA DE ACTIVIDADES Y HORAS APROXIMADAS DE TRABAJO - MARIO ARAYA

Rango de fechas	Actividades	Horas aproximadas
24 de Febrero - 2 de Marzo	Creación de niveles con arte de píxeles. Mapeo de los niveles a coordenadas hexadecimales para la lógica.	5
3 de Marzo - 9 de Marzo	Corrección de errores gráficos y de desplazamiento. Revisión del mapeo en la parte gráfica.	7
10 de Marzo - 13 de Marzo	Modificación de textos finales en pantalla. Mostrar el nivel actual en pantalla. Modificación de los textos de menús. Trabajo en la documentación final.	5

Cuadro V
TABLA DE ACTIVIDADES Y HORAS APROXIMADAS DE TRABAJO - FABIÁN RAMÍREZ

Rango de fechas	Actividades	Horas aproximadas
24 de Febrero - 3 de Marzo	Movimiento y renderización del jugador.	6
3 de Marzo - 11 de Marzo	Renderización de las paredes del mapa y la meta.	6
11 de marzo	Detección de colisiones.	2
11 de marzo	Implementación de funcionalidades adicionales como pausar y resetear el juego.	2

Cuadro VI
TABLA DE ACTIVIDADES Y HORAS APROXIMADAS DE TRABAJO - WAJIB ZAGLUL

Rango de fechas	Actividades	Horas aproximadas
24 de Febrero - 3 de Marzo	Análisis de requerimientos. Reuniones para definición de roles. Investigación sobre implementación	5
4 de Marzo - 6 de Marzo	Creación de Bootloader y Makefile	5
10 de marzo	Solución de errores de bootloader a la hora de correr desde USB en VirtualBox	2
12 de marzo	Solución a la imposibilidad de bootear desde usb en PC..	4