UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

LABORATORIO 1 DE ORGANIZACIÓN DE COMPUTADORES

DANY RUBIANO JIMENEZ

Profesores: Felipe Garay Pérez

Santiago - Chile

5 de enero de 2016

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

CAPÍTULO 1. INTRODUCCIÓN

1.1 MOTIVACIÓN Y ANTECEDENTES

El Buscaminas es un juego muy popular que se ha desarrollado en varios sistemas operativos, con el objeto de aprender el uso del lenguaje de programación Mips, y del complemento de los contenidos vistos en la cátedra de la asignatura, se pretende implementar este juego en el simulador MARS para Mips.

1.2 OBJETIVOS

El objetivo principal de este laboratorio es implementar una versión del popular juego buscaminas para MIPS en el simulador MARS, teniendo en cuenta todas las características de este juego, incluyéndole además otras funcionalidades. A través de ello, se pretende estudiar y aplicar todas las funcionalidades que ofrece este lenguaje y las características que se pueden desarrollar a través del simulador

1.3 ORGANIZACIÓN DEL DOCUMENTO

Este informe se desarrolla en varios capítulos, en el cual se incluyen un Marco Teórico, donde se muestran algunos de los conceptos emprendidos, un capitulo para el desarrollo, en donde se muestra como se va gestando la idea para la realización del juego, y por ultimo se incluye un capitulo para las conclusiones, en donde se da a conocer la síntesis de la experiencia en el desarrollo del laboratorio.

CAPÍTULO 2. MARCO TEÓRICO

2.0.1 Lenguaje Ensamblador

El lenguaje ensamblador, es un lenguaje de programación de bajo nivel para los computadores, microprocesadores, microcontroladores y otros circuitos integrados programables. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura dada de CPU y constituye la representación más directa del código máquina específico para cada arquitectura legible por un programador. Esta representación es usualmente definida por el fabricante de hardware, y está basada en los mnemónicos que simbolizan los pasos de procesamiento (las instrucciones), los registros del procesador, las posiciones de memoria y otras características del lenguaje. Un lenguaje ensamblador es por lo tanto específico de cierta arquitectura de computador física (o virtual). Esto está en contraste con la mayoría de los lenguajes de programación de alto nivel, que idealmente son portátiles. Un programa utilitario llamado ensamblador es usado para traducir sentencias del lenguaje ensamblador al código de máquina del computador objetivo. El ensamblador realiza una traducción más o menos isomorfa (un mapeo de uno a uno) desde las sentencias mnemónicas a las instrucciones y datos de máquina. Esto está en contraste con los lenguajes de alto nivel, en los cuales una sola declaración generalmente da lugar a muchas instrucciones de máquina. Muchos sofisticados ensambladores ofrecen mecanismos adicionales para facilitar el desarrollo del programa, controlar el proceso de ensamblaje, y la ayuda de depuración. Particularmente, la mayoría de los ensambladores modernos incluyen una facilidad de macro (descrita más abajo), y se llaman macro ensambladores. Fue usado principalmente en los inicios del desarrollo de software, cuando aún no se contaba con potentes lenguajes de alto nivel y los recursos eran limitados. Actualmente se utiliza con frecuencia en ambientes académicos y de investigación, especialmente cuando se requiere la manipulación directa de hardware, alto rendimiento, o un uso de recursos controlado y reducido. También es utilizado en el desarrollo de controladores de dispositivo (en inglés, device drivers) y en el desarrollo de sistemas operativos, debido a la necesidad del acceso directo a las instrucciones de la máquina. Muchos dispositivos programables (como los microcontroladores) aún cuentan con el ensamblador como la única manera de ser manipulados. (Wikipedia, s.f.-b).

2.0.2 Simulador Mars Mips

El simulador MARS Mips, es un editor de lenguaje ensamblador, ensamblador, simulador, y depurador combinado para el procesador MIPS. Fue desarrollado por Pete Sanderson y Kenneth Vollmar en la Universidad Estatal de Missouri. El simulador MARS está escrito en Java 1.4.2 , utilizando el estándar técnicas de interacción humano-computadora a través de paquetes swing y AWT. (Dr. Kenneth Vollmar, s.f.).

2.0.3 Buscaminas

Buscaminas es un videojuego para un jugador inventado por Robert Donner en 1989. El objetivo del juego es despejar un campo de minas sin detonar ninguna. El juego ha sido programado para muchos sistemas operativos, pero debe su popularidad a las versiones que vienen con Microsoft Windows desde su versión 3.1. El juego consiste en despejar todas las casillas de una pantalla que no oculten una mina. Algunas casillas

tienen un número, el cual indica la cantidad de minas que hay en las casillas circundantes. Así, si una casilla tiene el número 3, significa que de las ocho casillas que hay alrededor (si no es en una esquina o borde) hay 3 con minas y 5 sin minas. Si se descubre una casilla sin número indica que ninguna de las casillas vecinas tiene mina y éstas se descubren automáticamente. Si se descubre una casilla con una mina se pierde la partida. Se puede poner una marca en las casillas que el jugador piensa que hay minas para ayudar a descubrir las que están cerca. (Wikipedia, s.f.-a).

CAPÍTULO 3. DESARROLLO

Como primer funcionalidad del juego se piensa en la generación del tablero para mostrarlo a través del uso del display. Para ello, se confeccionó los diferentes gráficos que se necesitaron para dicha tarea, entre ellos destaca, un cuadro que representa el estado inicial de cada casilla, así mismo un cuadro que representa una mina, y un cuadro que representa la casilla vacía. Como entre las funcionalidades que se piden para el juego, se encuentra que se debe indicar con un número la cantidad de minas que hay en las casillas vecinas, entonces de la misma forma, se generaron gráficos para la representación de los cuadros con números. Se decidió en base a prueba y error, considerar finalmente cada cuadrado de 13 x 13 píxeles y así generar el tablero. Se utilizó el conversor de imágenes, para obtener los RGB de cada píxel, y así mediante una función dibujar estas en el display. Así mismo, se desarrollaron gráficos para acompañar la estructura del juego, considerando no sólo el tablero, sino también otros acompañante tales como un logo, la puntuación, mensajes de derrota o victoria, entre otros.

Un ejemplo de la generacion completa del tablero se muestra en la figura 3.1.

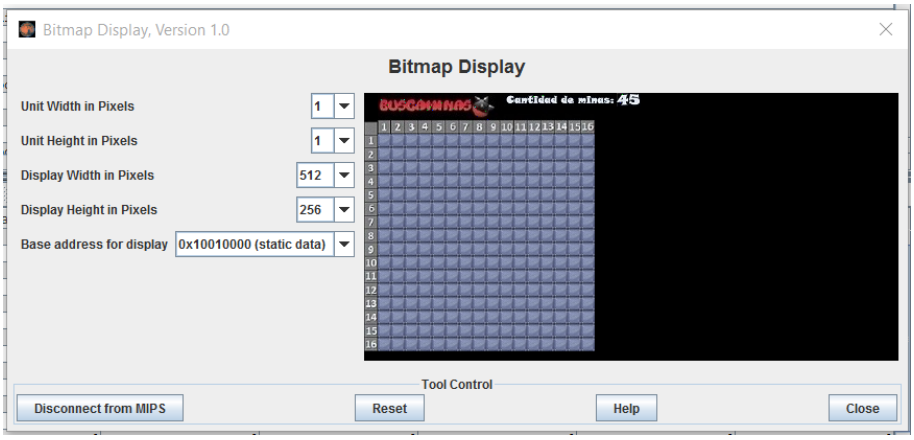

Figura 3-1: Tablero Generado con los gráficos utilizados

Para la representación del tablero dentro de los algoritmos, se utiliza un arreglo de tamño 256, puesto que este es el numero de casillas máximo correspondiente a un tablero de 16x16. Luego se procedió entonces a generar una algoritmo random para que dada cierta cantidad de minas, poder esparcirlas a lo largo del tablero.

Se tomo como opción, desarrollar una mayor interacción con el usuario, por lo que en una de las opciones del juego el usuario puede ingresar la matriz que desea, comprendiendo valores entre 4 y 16, debido a que menor que ese numero, resulta tal vez ilógico el jugar. Así mismo, se le pide al usuario que ingrese la cantidad de minas que desea, que puede comprender valores entre 1 y la cantidad de casillas dividida en tres, *ej: si es un tablero de 16 x 16, el total de minas puede comprender valores entre 1 y 85.*

Después de ello, se procede a implementar una función que cuenta en el tablero todas las casillas vecinas a una mina, para de esta forma cumplir con la funcionalidad pedida.

En continuación, se establecen los parámetros del juego, donde se interactúa con el usuario, existiendo

dos opciones, una para marcar una casilla, en donde el usuario marca la casilla donde cree que hay una mina, o bien la otra opción que corresponde a setear una casilla. Con este último, se establecen las condiciones del juego, en donde si la casilla seleccionada corresponde a una mina, se pierde el juego, o en caso contrario se aumenta la puntuación, hasta que solo que den minas en el tablero, si esto es así se gana el juego.

Un ejemplo de la interacción del juego con el usuario se muestra en la figura 3.2.

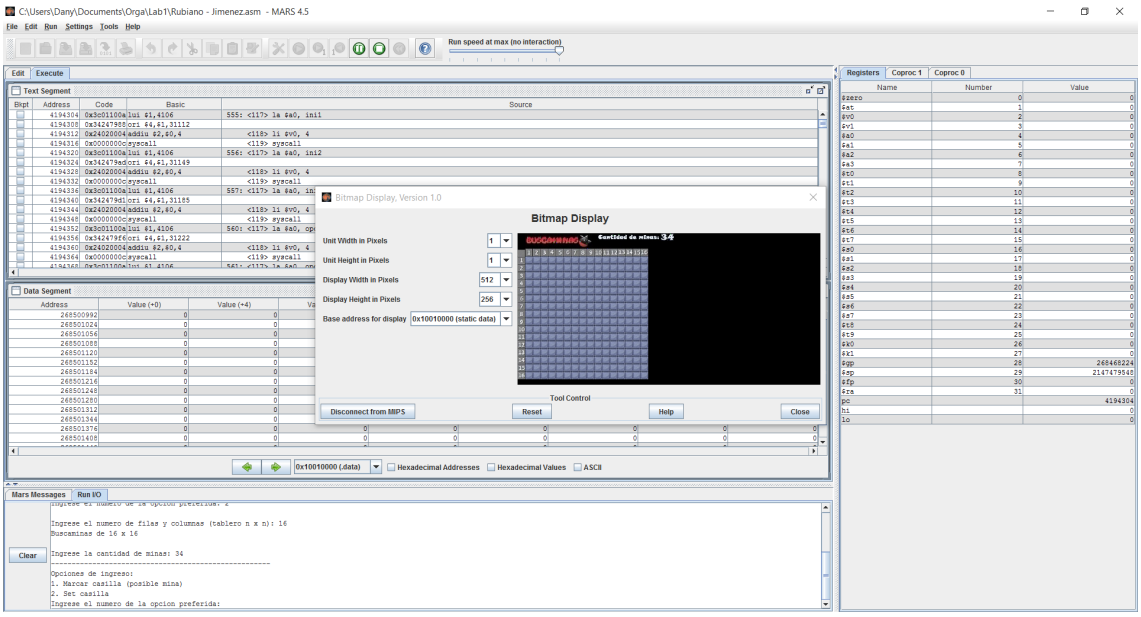

Figura 3-2: Ejemplo de interacción con el juego

CAPÍTULO 4. CONCLUSIONES

Después de el desarrollo de este laboratorios, se puede concluir lo siguiente:

En primer lugar resulto complicado emprender el camino para el desarrollo de este laboratorio, debido a la dificultad que resulta el realizar tareas complejas a través de un lenguaje ensamblador, ya que este es un lenguaje de bajo nivel y tiene funciones limitadas, por lo que el tiempo dedicado para desarrollar las funcionalidades pedidas para el juego, fue largo y arduo.

Se puede afirmar que no se cumplieron los objetivos en su totalidad, ya que no se pudo implementar varias de las funcionalidades exigidas, tales como leer un archivo con las minas, incluir la puntuación en el display, y dada la selección de una casilla vacías, obtener todas las casillas vacías alrededor de esta.

Resulto complicado para un programa de esta complejidad, el trabajar solo con las variables que dispone el simulador, por lo que se tuvo que optimizar el código y buscar maneras alternativas para guardar los datos a lo largo de los algoritmos. La eficiencia de los algoritmos desarrollados no es la ideal y no siempre cumplen con la modularidad, pero entregan el resultado esperado.

Este laboratorio sirvió de experiencia para trabajar con un lenguaje ensamblador MIPS, así mismo complementar lo aprendido en la cátedra de la arquitectura de MIPS, comprendiendo el uso del procesador. A través de este, se puede afirmar que se tiene un conocimiento básico para el desarrollo de otros laboratorios futuros.

CAPÍTULO 5. BIBLIOGRAFÍA

Dr. Kenneth Vollmar, D. P. S. (s.f.). Mars: an education-oriented mips assembly language simulator. http://courses.missouristate.edu/KenVollmar/mars/fp288-vollmar.pdf.

Garay, F. (2013, Diciembre). Juegos en mips. http://www.udesantiagovirtual.cl/moodle/file.php/2753/Laboratorio/apuntes/juegos_mips.pdf.

Wikipedia. (s.f.-a). Buscaminas. https://es.wikipedia.org/wiki/Buscaminas.

Wikipedia. (s.f.-b). Lenguaje ensamblador. https://es.wikipedia.org/wiki/Lenguaje_ensamblador.

y Erika Rosas, N. H. (2013, noviembre). Tutorial de mars: bitmap display. http://www.udesantiagovirtual.cl/moodle/file.php/2753/Laboratorio/apuntes/tutorial_display.pdf.