Índice

1. Introducción

Este trabajo consiste en estudiar como es que funcionan las funciones implementadas en asembler de MIPS32 y su performance. Además se mostrara como es que se estructura la memoria del programa al momento de llamar a las funciones implementadas en asembler.
Para lograr esto se implementó el desarrollo del conocido juego El juego de la vida el cual se encuentra gran parte hecho con C y desde este código se invocara a distintas sentencias implementadas en asembler.

2. Diseño e Implementación

2.1. Análisis de la linea de comandos

La aplicación acepta ser ejecutada con los siguientes parámetros:

-v Versión.Devuelve por salida estándar la versión del programa.

-h Help. Devuelve por salida estándar la información básica sobre como ejecutar el programa.

-o Output. Parámetro opcional, [-o "nombreArchivo"] permite especificarle al programa el nombre de los archivos para generar a la salida. De no especificarse ninguno, el nombre es "default"

-p Print. Permite indicar al programa que no se genere archivo a la salida.

2.2. Desarrollo del código fuente

Definir nuestros .c y .h

3. Proceso de compilación

Se cuenta con un archivo Makefile que indica las reglas de compilación ejecutadas por el comando make. El código fuente se compila ejecutando el comando make con alguna de las posibles reglas. En todos los casos se compila con los flags de warnings y de debbugging prendidos.

make Compila todo el programa hecho en lenguaje C

make mips Compila todo el programa hecho en C con la función "vecinosïmplementada en assembler

make clean Borra todos los archivos compilados del directorio y también los archivos .pbm generados por el programa

definir alguno mas text text text

4. Portabilidad

Pese a contener fragmentos en assembler MIPS32, es necesario que la implementación desarrollada provea un grado mínimo de portabilidad.
Para satisfacer esto, el programa deberá proveer dos versiones de conway(), incluyendo la versión MIPS32, pero también una versión C, pensada para dar soporte genérico a aquellos entornos que carezcan de una versión mas especıfica.

5. Casos de prueba

Para realizar las pruebas de esta aplicación se utilizaron 3 archivos bases. Donde cada uno indica como es inicializada la matriz del programa.
Cada linea del archivo corresponde con una coordenada de una fila y una columna en este respectivo orden, se asume que el valor inicial de la matriz es 0 y solo se establecerán en valor 1 aquellos indicados por el archivo en cuestión

- pento
- sapo
- glider

A continuación se procede a indicar el contenido de cada archivo y su respectivo resultado.
Pento:

X	Y
3	5
3	6
4	4
3	6
4	5
5	5

Cuadro 1: Tabla de estados iniciales en 1

Lo que nos genera un tablero inicial tal como se ve en la siguiente imagen

Sapo:

X	Y
5	3
5	4
5	5
4	4
4	5
5	6

Cuadro 2: Tabla de estados iniciales en 1

Lo que nos genera un tablero inicial tal como se ve en la siguiente imagen

Glider:

X	Y
5	3
5	4
5	5
3	4
4	5

Cuadro 3: Tabla de estados iniciales en 1

Lo que nos genera un tablero inicial tal como se ve en la siguiente imagen