Trabajo Práctico 1
Conjunto de instrucciones MIPS

Marco Rodrigo Albanesi, *Padrón Nro. 86.063*
fiuba@mrod.com.ar

Natalia Nayla Alvarez Ledesma, *Padrón Nro. 90.928*
natalia.nayla.alvarez@gmail.com

1er. Cuatrimestre del 2020
66.20 Organización de Computadoras – Práctica: Jueves
Facultad de Ingeniería, Universidad de Buenos Aires

11 de junio de 2020

Resumen

Este trabajo práctico tiene como objetivo familiarizarse con el conjunto de instrucciones MIPS32 y el concepto de ABI explicado en clases, desarrollando un programa portable que resuelva el problema del juego de la vida planteado por Conway.

1. Enunciado

66:20 Organización de Computadoras
Trabajo práctico 1: conjunto de instrucciones MIPS

1. Objetivos

Familiarizarse con el conjunto de instrucciones MIPS32 y el concepto de ABI[1], escribiendo un programa portable que resuelva el problema descripto en la sección 6.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes.

Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 9), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo, y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

4. Recursos

Usaremos el programa `qemu` [2] para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión reciente del sistema operativo Debian [3].

[1] Application Binary Interface

5. Introducción

El "Juego de la Vida" de Conway es un autómata celular, diseñado por el matemático británico John Conway [5] en 1970 [6], y si bien su funcionamiento es simple, computacionalmente es equivalente a una máquina de Turing [7]. Se trata básicamente de una grilla en principio infinita, en cada una de cuyas celdas puede haber un organismo vivo (se dice que la celda está viva, o encendida) o no, en cuyo caso se dice que está muerta o apagada. Se llama "vecinos" de una celda a las ocho celdas adyacentes a ésta. Esta matriz evoluciona en el tiempo en pasos discretos, o estados, y las transiciones de las celdas se realizan de la siguiente manera:

- Si una celda tiene menos de dos o más de tres vecinos encendidos, su siguiente estado es apagado.
- Si una celda encendida tiene dos o tres vecinos encendidos, su siguiente estado es encendido.
- Si una celda apagada tiene exactamente tres vecinos encendidos, su siguiente estado es encendido.
- Todas las celdas se actualizan simultáneamente.

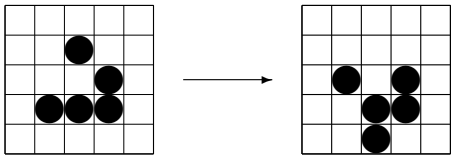

Figura 1: Ejemplo de transición entre estados

6. Programa

Se trata de una versión en lenguaje C del "Juego de la Vida". El programa recibirá por como argumentos tres números naturales, i, M y N, y el nombre de un archivo de texto con coordenadas en una matriz de MxN, y escribirá i archivos .PBM [8] [2] representando i estados consecutivas del "Juego de la Vida" en una matriz de MxN, tomando como celdas 'vivas' iniciales las que están en las coordenadas del archivo de entrada (el primer estado a representar es el inicial). De haber errores, los mensajes de error deberán salir exclusivamente por `stderr`. Si la corrida fue exitosa, usar el

[2] Los nombres de los archivos deberán ser del formato `[outputprefix]_NNN.pbm`, con `NNN` representando números de orden con ceros a la izquierda: `pref_001.pbm`, `pref_002.pbm`, etc

programa `ffmpeg`[9] para hacer un video estilo "stop motion"[10] (con el paso 3 alcanza).

6.1. Condiciones de contorno

Dados los problemas que acarrearía tratar con una matriz infinita, se ha optado por darle un tamaño limitado. En este caso, para las filas y columnas de los 'bordes' de la matriz, hay dos maneras básicas de calcular los 'vecinos':

1. **Hipótesis del mundo rectangular:** Las posiciones que caerían fuera de la matriz se asumen apagadas. Sencillamente no se computan.
2. **Hipótesis del mundo toroidal:** La fila $M-1$ pasa a ser vecina de la fila 0, y la columna $N-1$ pasa a ser vecina de la columna 0. Entonces, el vecino superior de la celda $[0, j]$ es el $[M-1, j]$, el vecino izquierdo de la celda $[i, 0]$ es el $[i, N-1]$, y viceversa; de esta manera, nunca nos salimos de la matriz. Esta es la opción más interesante, y la que debe usar el programa.

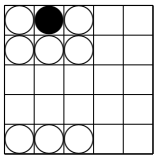

Figura 2: Ejemplo de mundo toroidal: la celda (0,1) y sus vecinos.

6.2. Comportamiento deseado

Primero, usamos la opción `-h` para ver el mensaje de ayuda:

```
$ conway -h
Uso:
  conway -h
  conway -V
  conway i M N inputfile [-o outputprefix]
Opciones:
  -h, --help     Imprime este mensaje.
  -V, --version  Da la versión del programa.
  -o  Prefijo de los archivos de salida.
Ejemplos:
 conway 10 20 20 glider -o estado
 Representa 10 iteraciones del Juego de la Vida en una matriz de 20x20,
 con un estado inicial tomado del archivo ‘‘glider’’.
 Los archivos de salida se llamarán estado_n.pbm.
 Si no se da un prefijo para los archivos de salida,
```

```
 el prefijo será el nombre del archivo de entrada.
```

Ahora usaremos el programa para generar una secuencia de estados del Juego de la Vida.

```
$ conway 5 10 10 glider -o estado
Leyendo estado inicial...
Grabando estado_1.pbm
Grabando estado_2.pbm
Grabando estado_3.pbm
Grabando estado_4.pbm
Grabando estado_5.pbm
Listo
```

El formato del archivo de entrada es de texto, con los números de fila y columna separados por espacios, a una celda ocupada por línea. Ejemplo: si el archivo `glider` representa un planeador en el centro de una grilla de $10x10$, se verá de la siguiente manera:

```
$ cat glider
5 3
5 4
5 5
3 4
4 5
```

El programa deberá retornar un error si las coordenadas de las celdas encendidas están fuera de la matriz ($[0..M-1], [0..N-1]$), o si el archivo no cumple con el formato.

7. Implementación

El programa a implementar deberá satisfacer algunos requerimientos mínimos, que detallamos a continuación.

7.1. Portabilidad

Pese a contenter 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 más específica.

7.2. API

Gran parte del programa estará implementada en lenguaje C. Sin embargo, la función `vecinos()` estará implementada en assembler MIPS32, para proveer soporte específico en nuestra plataforma principal de desarrollo, MIPS32.

El programa en C deberá interpretar los argumentos de entrada, pedir la memoria necesaria para la matriz de estado A, popular la matriz con los contenidos del archivo de entrada, y computar el siguiente estado para cada celda valiéndose de la siguiente función:

```
unsigned int vecinos(unsigned char *a,
                     unsigned int i, unsigned int j,
                     unsigned int M, unsigned int N);
```

Donde `a` es un puntero a la posición $[0, 0]$ de la matriz, i y j son la fila y la columna respectivamente del elemento cuyos vecinos queremos calcular, y M y N son las cantidades de filas y columnas de la matriz A. El valor de retorno de la función `vecinos` es la cantidad de celdas vecinas que están encendidas en el estado actual. Los elementos de A pueden representar una celda cada uno, aunque para reducir el uso de memoria podrían contener hasta ocho cada uno. Después de computar el siguiente estado para la matriz A, el programa deberá escribir un archivo en formato PBM [8] representando las celdas encendidas con color blanco y las apagadas con color negro [3].

7.3. ABI

El pasaje de parámetros entre el código C (`main()`, etc) y la rutina `vecinos()`, en assembler, deberá hacerse usando la ABI explicada en clase: los argumentos correspondientes a los registros `$a0-$a3` serán almacenados por el *callee*, siempre, en los 16 bytes dedicados de la sección "function call argument area" [4].

7.4. Algoritmo

El algoritmo a implementar es el algoritmo del Juego de la Vida de Conway[6], explicado en clase.

8. Proceso de Compilación

En este trabajo, el desarrollo se hará parte en C y parte en lenguaje Assembler. Los programas escritos serán compilados o ensamblados según el caso, y posteriormente enlazados, utilizando las herramientas de GNU

[3] Si se utiliza sólo un pixel por celda, no se podrá apreciar el resultado a simple vista. Pruebe haciendo que una celda sea representada por grupos de por ejemplo 16x16 pixels.

disponibles en el sistema NetBSD utilizado. Como resultado del enlace, se genera la aplicación ejecutable.

9. Informe

El informe deberá incluir [4]:

- Este enunciado;
- Documentación relevante al diseño e implementación del programa, incluyendo un diagrama del stack de la función `vecinos`;
- Corridas de prueba para diez iteraciones, en una matriz de $20x20$, de los archivos de entrada `glider`, `pento` y `sapo`, con los comentarios pertinentes;
- El código fuente completo, en formato digital.

10. Mejoras opcionales

- Una versión de terminal, que permita ver en tiempo real la evolución del sistema (y suprima los archivos de salida).
- Un editor de pantalla, de modo texto, a una celda por caracter. Esto permite experimentar con el programa, particularmente combinado con la versión de tiempo real.

11. Fecha de entrega

Primera entrega: Semana del 21 de Mayo.
Revisión: Semana del 28 de Mayo.
Última fecha de entrega: Semana del 4 de Junio.

Referencias

[1] GXemul, `http://gavare.se/gxemul/`.

[2] QEMU, `https://www.qemu.org/`

[3] Debian, the Universal Operating System, `https://www.debian.org/`.

[4] System V application binary interface, MIPS RISC processor supplement (third edition). Santa Cruz Operations, Inc.

[4] no incluir el enunciado en el .tex del informe, con agregar el PDF a la entrega alcanza

[5] John Horton Conway, [1937-2020], https://en.wikipedia.org/wiki/John_Horton_Conway.

[6] Juego de la Vida de Conway, http://es.wikipedia.org/wiki/Juego_de_la_vida.

[7] Máquina de Turing, implementada en el Juego de la Vida de Conway, http://rendell-attic.org/gol/tm.htm.

[8] http://netpbm.sourceforge.net/doc/pbm.html

[9] https://www.ffmpeg.org/

[10] https://lukecyca.com/2013/stop-motion-with-ffmpeg.html

2. Desarrollo

2.1. Diseño e implementación del programa

Se diseñó un programa en lenguaje C que implementa el juego de la vida de Conway, descripto en la sección 5 de este trabajo práctico. Esta desarrollada para que provea un grado mínimo de portabilidad, por lo que tiene dos versiones de conway(), una desarrollada enteramente en C y otra versión que tiene una función implementada en MIPS32.

2.1.1. Análisis de la línea de comandos

Las opciones válidas ingresadas por la línea de comandos como fue mencionado, puede ser:

```
-h, --help: Muestra la ayuda para ejecutar correctamente la
    aplicación.
-V, -- version: Muestra la versión del programa.
-o : define el prefijo de los archivos de entrada, en caso de no
    especificarse se usa por defecto el nombre del archivo de
    entrada.
-d, --display:   Muestra el estado de cada iteración por la
    salida estándar.
-n, --novideo   Omite la generación de un video en base a los
    archivos de salida.
```

2.1.2. Desarrollo del código fuente

Se diseño un programa en C que implementa el juego de la vida de Conway, que si el ingreso de parámetros es inválida, muestra por pantalla el mensaje explicativo mencionado arriba. La estructura del programa es de la siguiente manera:

Funciones básicas Posee el main.c donde se declaran las funciones para el correcto funcionamiento de este trabajo:

```
        void mostrar_ayuda()
        int abrirArchivo()
        int cerrarArchivo()
        int cargarMatriz()
        int guardarEstado()
        void mostrarEstado()
        int parsearParametros()
        int generarVideo()
```

Uso de bibliotecas Se hace uso de la función `getopt_long()` de la biblioteca `getopt.h`. Dicha función provee una forma simple de procesar cada opción que es leída, extrayendo los argumentos de cada una. En caso de que no se encuentre alguna opción, se utiliza su valor por defecto según las especificaciones del

trabajo.

Funciones especiales La función `vecinos()` se encuentra implementada en vecinos.c, completamente en C y en vecinos.s completamente en MIPS32. Ambas funciones realizan lo mismo y por ende, esto brinda portabilidad a la aplicación.

La implementación en MIPS al ser una función hoja no necesita crear la zona de Argument Building Area y tampoco necesita persistir en su Saved Register Area el valor del registro ra. Además, como todas las variables que necesita se pueden ubicar en los registros temporales t0-t9 no se necesita guardar nada en la Local and Temporary Area.

Al no escribir en otros registros y como no se llama a otra función que puedar sobreescribir los que está en uso, el stack solo necesita capacidad para guardar el valor de los registros gp y fp. Es decir 8 bytes:

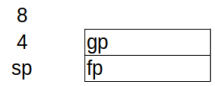

Figura 1: Diagrama del stack de vecinos

Funcionamiento de la aplicación

Una vez realizada la validación de las opciones y carga del archivos de entrada, usando la ABI explicada en clases donde los argumentos correspondientes a los registros $a0-a3$ serán almacenados por el callee, siempre, en los 16 bytes dedicados de la sección "function call argument area", se utiliza la función de vecino para ver las celdas adyacentes y generar un nuevo estado usando las iteraciones dadas en el argumento de entrada. Se generan los n archivos de salidas con el prefijo también dado o un default en caso de no especificarse. Estos archivos de salida son una foto del estado en que se encuentra en esa iteración, además genera un .mp4 mostrando el mismo en un vídeo.

El código fuente se encuentra en el apéndice, tanto en lenguaje C como en MIPS32.

3. Compilación del programa

Para compilar el programa sencillamente tenemos utilizar su makefile desde el directorio donde se descomprimió el entregable.

```
$ make
cc src/main.c src/vecinos.S -o conway -lm -O0 -Wall -Werror -
    pedantic -pedantic-errors
```

Esto compilará el programa usando la implementación de vecinos según la arquitectura en la que se encuentre (C para x86 y assembly para MIPS). En caso de querer compilar usando la implementación en C para MIPS hay que ejecutar:

```
$ make clean
$ C_IMPL=1 make
```

```
cc src/main.c src/vecinos.c -o conway -lm -O0 -Wall -Werror -
    pedantic -pedantic-errors
```

4. Pruebas

En el caso uno, tomaremos como prueba la ejecución con los siguientes parámetros:

```
Prueba 1: generación del tablero con archivo glider, 10 veces de
    20x20.
$ ./conway 10 20 20  glider -n
```

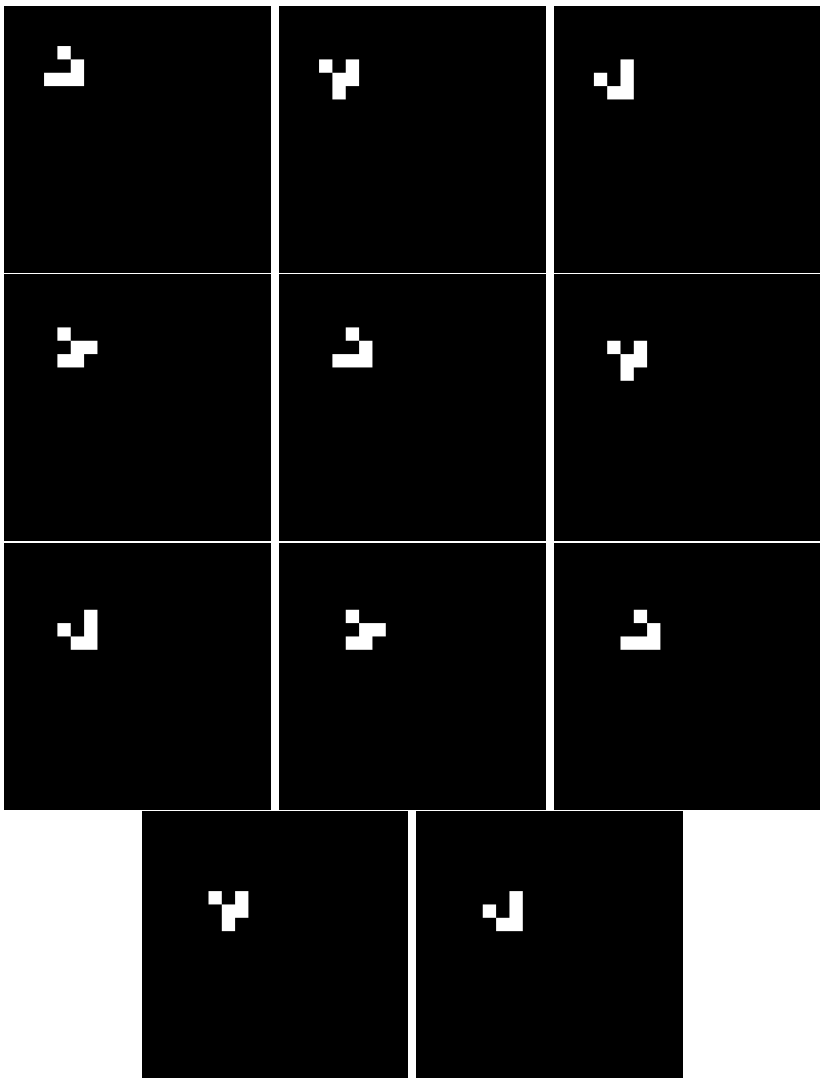

Figura 2: Caso de prueba Glider

Ahora realizamos lo mismo, solo cambiando el archivo de entrada por sapo:

```
Prueba 2: generación del tablero con archivo sapo, 10 veces de 20
    x20.
$ ./conway 10 20 20  sapo -n
```

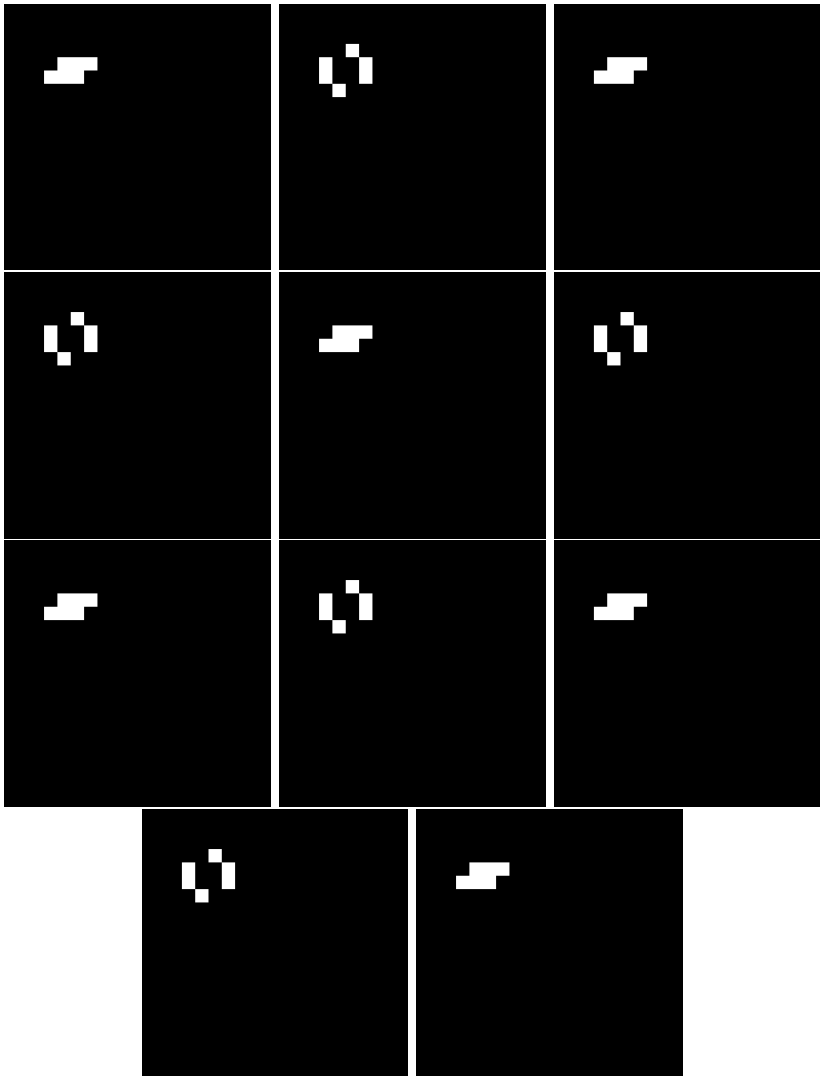

Figura 3: Caso de prueba Sapo

Continuamos con el archivo pento, usando los mismo parámetros.

```
Prueba 3: generación del tablero con archivo pento, 10 veces de
    20x20.
$ ./conway 10 20 20  pento -n
```

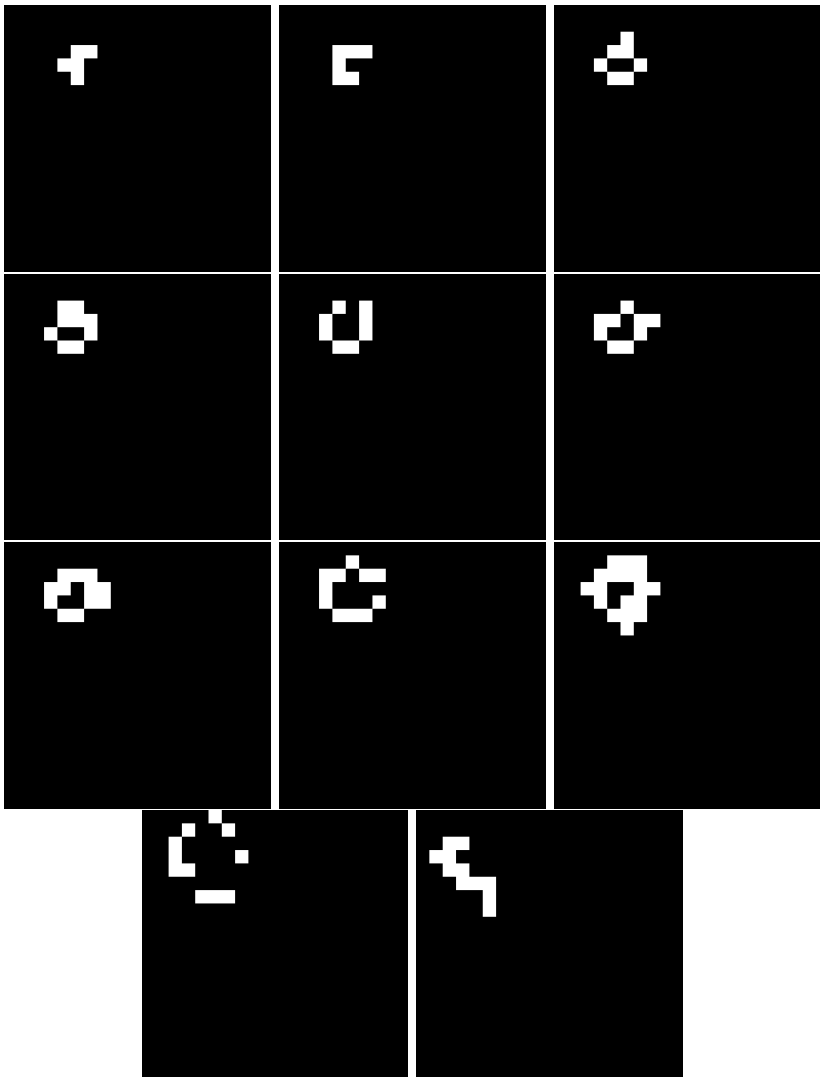

Figura 4: Caso de prueba Pento

Corroboramos los casos en donde no tiene parámetros de entrada:

```
Prueba 4: sin parámetros de entrada
$ ./conway

  PROGRAM OUTPUT:

Uso:
        conway -h
        conway -V
        conway i M N inputfile [-o outputprefix] [-d] [-n]
Opciones:
        -h, --help      Imprime este mensaje.
        -V, --version   Da la version del programa.
```

```
        -d, --display   Muestra el estado de cada iteración por
    la salida estandar.
        -o              Prefijo de los archivos de salida.
        -n, --novideo   Omite la generación de un video en base a
     los archivos de salida.
Ejemplos:
 conway 10 20 20 glider -o estado
 Representa 10 iteraciones del Juego de la Vida en una matriz de
    20x20,
  con un estado inicial tomado del archivo glider.
  Los archivos de salida se llamaran estado_n.pbm.
  Si no se da un prefijo para los archivos de salida,
  el prefijo sera el nombre del archivo de entrada.
```

Por ultimo probamos mostrar el estado por la salida estándar usando el parámetro -d y lo comparamos con la imagen generada:

```
Prueba 5: Salida estándar
$ ./conway 0 18 18 xkcd -d
/-----------------------------------\
|                                   |
|                                   |
|                                   |
|                                   |
|                                   |
|              ()()()               |
|              ()  ()               |
|              ()  ()               |
|                ()                 |
|          ()  ()()()               |
|            ()  ()  ()             |
|                ()    ()           |
|              ()  ()               |
|              ()  ()               |
|                                   |
|                                   |
|                                   |
|                                   |
\-----------------------------------/
```

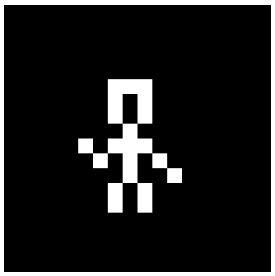

Figura 5: Caso de prueba stdout

5. Conclusiones

En este trabajo practico se puso en practica lo visto en clases por lo que nos pareció muy útil para ver usos en la portabilidad del mismo y el manejo de la arquitectura MIPS32. Además, ver las diferencias entre programar en un sistema operativo linux y en otro emulado en Qemu.

Referencias

[1] Kernighan, B. W. - Ritchie, D. M. - *C Programming Language* - 2nd edition - Prentice Hall - 1988.

[2] Patton, R. - *Software Testing* - 2nd edition - Sams Indianapolis, IN, USA 2005.

[3] *Apuntes del curso 66.20 Organización de Computadoras* - Cátedra Hamkalo - Facultad de Ingeniería de la Universidad de Buenos Aires.

[4] *GNU Make* - https://www.gnu.org/software/make/

[5] *GNU Gcc* - https://gcc.gnu.org/

[6] Stop-motion with ffmpeg, https://lukecyca.com/2013/stop-motion-with-ffmpeg.html

[7] QEMU, https://www.qemu.org/

[8] xkcd - RIP John Conway, https://xkcd.com/2293/

6. Apéndice: Código fuente

6.0.1. main.c

```
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <getopt.h>
#include <math.h>
#include <unistd.h>

#include "vecinos.h"

#define ERROR -1
#define OK 0

void mostrar_ayuda() {
    printf("%s", "Uso:\n\t"
            "conway␣-h\n\t"
            "conway␣-V\n\t"
            "conway␣i␣M␣N␣inputfile␣[-o␣outputprefix]␣[-d]␣[-n]␣\
    n"
            "Opciones:\n\t"
            "-h,␣--help\tImprime␣este␣mensaje.\n\t"
            "-V,␣--version␣\tDa␣la␣version␣del␣programa.\n\t"
            "-d,␣--display\tMuestra␣el␣estado␣de␣cada␣iteración␣
    por␣la␣salida␣estandar.\n\t"
            "-o\t\tPrefijo␣de␣los␣archivos␣de␣salida.\n\t"
            "-n,␣--novideo\tOmite␣la␣generación␣de␣un␣video␣en␣
    base␣a␣los␣archivos␣de␣salida.\n"
            "Ejemplos:\n"
            "␣conway␣10␣20␣20␣glider␣-o␣estado\n"
            "␣Representa␣10␣iteraciones␣del␣Juego␣de␣la␣Vida␣en␣
    una␣matriz␣de␣20x20,\n␣"
            "␣con␣un␣estado␣inicial␣tomado␣del␣archivo␣glider.\n␣
    "
            "␣Los␣archivos␣de␣salida␣se␣llamaran␣estado_n.pbm.\n␣
    "
            "␣Si␣no␣se␣da␣un␣prefijo␣para␣los␣archivos␣de␣salida
    ,\n␣"
            "␣el␣prefijo␣sera␣el␣nombre␣del␣archivo␣de␣entrada.\n
    ");
}


int abrirArchivo(FILE** file, const char *nombre, const char *
    modo) {
    *file = fopen(nombre, modo);
    if (*file == NULL) {
```

```
        fprintf(stderr, "Error␣abriendo␣el␣archivo␣'%s':␣",
    nombre);
        perror(NULL);
        return ERROR;
    }
    return OK;
}

int cerrarArchivo(FILE* file) {
    if (file == NULL)
        return ERROR;

    if (fclose(file) == ERROR) {
        perror("Error␣cerrando␣archivo");
        return ERROR;
    }
    return OK;
}

int cargarMatriz(unsigned char* matriz, char* entrada, unsigned
    int filas, unsigned int columnas) {
    int status = OK;
    FILE* archivo;
    status = abrirArchivo(&archivo, entrada, "r");

    if (status != ERROR) {
        for (int idx = 0; idx < filas * columnas; idx++)
            matriz[idx] = 0;

        int i, j;

        do {
            if (fscanf(archivo, "%d␣%d\n", &i, &j) != 2) {
                if (! feof(archivo) || !ferror(archivo)) {
                    fprintf(stderr, "Error␣de␣formato␣en␣el␣
    archivo␣de␣entrada\n");
                    status = ERROR;
                }
            } else {
                if (i < 0 || i >= filas || j < 0 || j >= columnas
    ) {
                    fprintf(stderr, "Coordenadas␣fuera␣de␣la␣
    matriz:␣(%d;%d)\n", i, j);
                    status = ERROR;
                } else
                    matriz[i * columnas + j] = 1;
            }
        } while (status != ERROR && ! feof(archivo));

        if (ferror(archivo)) {
```

```
            fprintf(stderr, "Error␣leyendo␣el␣archivo␣'%s':␣",
    entrada);
            perror(NULL);
            status = ERROR;
        }
    }

    cerrarArchivo(archivo);

    return status;
}

int guardarImagen(unsigned char* matriz, unsigned int filas,
    unsigned int columnas, char* salida, unsigned char escala) {
    int status = OK;
    FILE* archivo;
    status = abrirArchivo(&archivo, salida, "w");

    if (status != ERROR) {
        if (fprintf(archivo, "P4\n%d␣%d\n", columnas * escala,
    filas * escala) > 0) {
          for (int i = 0; i < filas && status != ERROR; i++) {
              for (int h = 0; h < escala && status != ERROR; h++)
     {
          for (int j = 0; j < columnas && status != ERROR; j++) {
            char c = matriz[i * columnas + j] ? 0 : 255;

            for (int w = 0; w < escala && status != ERROR; w+=8)
                status = fputc(c, archivo) < 0 ? ERROR : OK;
          }
              }
            }
        }

        if (ferror(archivo)) {
            fprintf(stderr, "Error␣escribiendo␣en␣el␣archivo␣'%s
    ':␣", salida);
            perror(NULL);
            status = ERROR;
        }
    }

    cerrarArchivo(archivo);
    return status;
}

void mostrarEstado(unsigned char* matriz, unsigned int filas,
    unsigned int columnas) {
    printf("/");
    for (int j = 0; j < columnas; j++) {
```

```
        printf("--");
    }

    printf("\\\n");
    for (int i = 0; i < filas; i++) {
        printf("|");

        for (int j = 0; j < columnas; j++) {
            if (matriz[i * columnas + j])
                printf("()");
            else
                printf("␣␣");
        }
        printf("|\n");
    }

    printf("\\");
    for (int j = 0; j < columnas; j++) {
        printf("--");
    }
    printf("/\n");
}

/**
Si una celda tiene menos de dos o mas de tres vecinos encendidos,
     su siguiente estado es apagado.
Si una celda encendida tiene dos o tres vecinos encendidos, su
    siguiente estado es encendido.
Si una celda apagada tiene exactamente tres vecinos encendidos,
    su siguiente estado es encendido.
* */
int siguiente(unsigned char** matriz, unsigned int filas,
    unsigned int columnas) {
    int status = OK;
  unsigned char* siguiente = malloc(filas * columnas);

  if (! siguiente) {
    fprintf(stderr, "Error␣reservando␣memoria\n");
    status = ERROR;
  }

  if (status == OK) {
    for (int i = 0; i < filas; i++) {
      for (int j = 0; j < columnas; j++) {
        unsigned char estado = (*matriz)[i * columnas + j];
        unsigned int v = vecinos(*matriz, i, j, filas, columnas);

        if (v < 2 || v > 3)
          estado = 0;
        else if (estado && (v == 2 || v == 3))
```

```
          estado = 1;
        else if (! estado && v == 3)
          estado = 1;

        siguiente[i * columnas + j] = estado;
      }
    }
  }

  if (siguiente) {
    free(*matriz);
    *matriz = siguiente;
  }

    return status;
}


int parsearParametros(int argc, char *argv[], unsigned int* pasos
    , unsigned int* filas, unsigned int* columnas, char** entrada
    , char** prefijoSalida, bool* mostrarEstados, bool* video) {
    int status = OK;

    static struct option longOpts[] = {
            { "help", no_argument, NULL, 'h' },
            { "version", no_argument, NULL, 'V' },
            { "o", required_argument, NULL, 'o' },
            { "display", no_argument, NULL, 'd' },
            { "novideo", no_argument, NULL, 'n' },
            { NULL, 0, NULL, 0 }
    };

    int c;
    unsigned short index = 0;
    do {
        c = getopt_long(argc, argv, "-hVo:dn", longOpts, NULL);

        switch (c) {
        case 1:
            if (index < 3) {
                int entero = 0;
                if (sscanf(optarg, "%d", &entero) != 1 || entero
    < 0)
                    status = ERROR;

                if (index == 0)
                    *pasos = entero;
                else if (index == 1)
                    *filas = entero;
                else if (index == 2)
```

```
                    *columnas = entero;

            } else if (index == 3)
                *entrada = optarg;
            else
                status = ERROR;

            if (status == ERROR) {
                if (index < 3)
                    fprintf(stderr, "Error␣en␣argumentos:␣'%s'␣no
    ␣es␣un␣entero\n", optarg);
                mostrar_ayuda();
                return ERROR;
            }
            index+=1;
            break;
        case 'V':
            printf("%s", "conway␣v0.2\n");
            return OK;
        case 'h':
            mostrar_ayuda();
            return OK;
        case 'o':
            *prefijoSalida = optarg;
            break;
        case 'd':
            *mostrarEstados = true;
            break;
        case 'n':
            *video = false;
            break;
        case '?':
            mostrar_ayuda();
            return ERROR;
        default:
            break;
        }
    } while (c != -1);

    if (index < 4) {
        mostrar_ayuda();
        return ERROR;
    }

    if (! *prefijoSalida)
        *prefijoSalida = *entrada;

    return status;
}
```

```
int generarVideo(char* prefijoSalida, unsigned char digitos) {
    unsigned int tamPrefijoSalida = strlen(prefijoSalida);

    char* salida = malloc(tamPrefijoSalida + 5); //extensión
    char* archivos = malloc(tamPrefijoSalida + 10); //"_%0Xd.pbm"

    if (! archivos || ! salida) {
        fprintf(stderr, "Error␣reservando␣memoria\n");
        return ERROR;
    }

    snprintf(salida, tamPrefijoSalida + 5, "%s.mp4",
    prefijoSalida);

    snprintf(archivos, tamPrefijoSalida + 10, "%s_%%0%dd.pbm",
    prefijoSalida, digitos);

    printf("Generando␣%s...\n", salida);

    int rv = execl("/usr/bin/ffmpeg", "ffmpeg", "-y", "-framerate
    ", "10", "-i", archivos, "-loglevel", "panic", "-r", "10",
    salida, (char*) NULL);

    if (salida)
        free(salida);

    if (archivos)
        free(archivos);

    return rv;
}


int main(int argc, char *argv[]) {
    unsigned int pasos=0, filas=0, columnas=0;
    char *entrada = 0;
    char *prefijoSalida = 0;
    bool mostrarEstados = false;
    bool video = true;

    int status = parsearParametros(argc, argv, &pasos, &filas, &
    columnas, &entrada, &prefijoSalida, &mostrarEstados, &video);
    unsigned char digitos = 3;

    if (pasos > 0) {
        digitos = floor(log10(pasos))+1;
        if (digitos > 9) {
            fprintf(stderr, "Usar␣una␣cantidad␣de␣pasos␣menor␣a␣
    10^9\n");
```

```
            status = ERROR;
        }
        if (digitos < 3)
            digitos = 3;
    }

    if (status != ERROR) {
        unsigned int salidalen = strlen(prefijoSalida) + digitos
    + 6; //6: len('_.pbm\0')
        char* salida = malloc(salidalen);

        char formatoSalida[] = "%s_%0?d.pbm";
        formatoSalida[5] = digitos + 48; //número a texto

        unsigned char* matriz = malloc(filas * columnas);

        if (! matriz || ! salida) {
            fprintf(stderr, "Error␣reservando␣memoria\n");
            status = ERROR;
        }

        if (status != ERROR)
            status = cargarMatriz(matriz, entrada, filas,
    columnas);

        if (status != ERROR) {
            if (mostrarEstados)
                mostrarEstado(matriz, filas, columnas);

            unsigned int escalaX = 1920 / columnas / 8;
            unsigned int escalaY = 1080 / filas / 8;
            unsigned char escala = escalaX < escalaY ? escalaX :
    escalaY;
            escala = escala > 0 ? escala * 8 : 8;

            snprintf(salida, salidalen, formatoSalida,
    prefijoSalida, 0);
      status = guardarImagen(matriz, filas, columnas, salida,
    escala);

            for (int paso = 1 ; paso <= pasos && status != ERROR;
     paso++) {
                status = siguiente(&matriz, filas, columnas);
                if (mostrarEstados)
                    mostrarEstado(matriz, filas, columnas);

                snprintf(salida, salidalen, formatoSalida,
    prefijoSalida, paso);
                status = guardarImagen(matriz, filas, columnas,
    salida, escala);
```

```
            }
        }

        if (matriz)
            free(matriz);

        if (salida)
            free(salida);

        if (status != ERROR && pasos > 0 && video) {
            if (generarVideo(prefijoSalida, digitos) < 0) {
                fprintf(stderr, "Error␣invocando␣a␣ffmpeg\n");
                status = ERROR;
            }
        }

    }

    return status;
}
```

6.0.2. vecinos.h

```
unsigned int vecinos(unsigned char *matriz, unsigned int i,
    unsigned int j, unsigned int filas, unsigned int columnas);
```

6.0.3. vecinos.c

```
#include "vecinos.h"

unsigned int vecinos(unsigned char *matriz, unsigned int i,
    unsigned int j, unsigned int filas, unsigned int columnas) {
    unsigned int v = 0;
    for (int y = -1; y <= 1; y++) {
        for (int x = -1; x <= 1; x++) {
            if (y || x) {
                int I = y+i;
                if (I < 0) I += filas;
                if (I == filas) I = 0;

                int J = x+j;
                if (J < 0) J += columnas;
                if (J == columnas) J = 0;

                v += matriz[I * columnas + J];
            }
        }
    }
```

```
    return v;
}
```

6.0.4. vecinos.S

```
#include <regdef.h>

.text
.abicalls
.align 2

/*
unsigned int vecinos(unsigned char *matriz, unsigned int i,
    unsigned int j, unsigned int filas, unsigned int columnas) {
    unsigned int v = 0;
    for (int y = -1; y <= 1; y++) {
        for (int x = -1; x <= 1; x++) {
            if (y || x) {
                int I = y+i;
                if (I < 0) I += filas;
                if (I == filas) I = 0;

                int J = x+j;
                if (J < 0) J += columnas;
                if (J == columnas) J = 0;

                v += matriz[I * columnas + J];
            }
        }
    }
    return v;
}

Stack:
   8|   fp
sp 0|   gp
*/

//labels para los parámetros
#define matriz a0
#define i a1
#define j a2
#define filas a3
#define columnas t0

//labes para variables temporales en registros
#define x t1
#define y t2
#define I t3
```

```
#define J t4
#define cmp t5
#define idx t6
#define val t7
#define addr t8
#define one t9

.globl  vecinos
.ent vecinos

vecinos:
    subu sp, sp, 8      //Stack de 8 bytes.
    sw gp, 4(sp)
    sw fp, 0(sp)
    move fp, sp

    lw columnas, 24(sp) //Obtener el quinto parámetro del stack
    de la funcion llamante

    move v0, zero       // v = 0
    li one, 1           // cte
    li y, -1            //y = -1

loopY:
    li x, -1            //x = -1

loopX:
    or cmp, y, x        //(y || x)
    beqz cmp, continue
    nop

    add I, y, i         // I = y+i;
    bltz I, IltZero     // if (I < 0)
    nop
    beq I, filas, IeqFilas  //(I == filas)
    nop

Icorrected:
    add J, x, j         //J = x+j;
    bltz J, JltZero     //if (J < 0)
    nop
    bge J, columnas, JeqColumnas    //if (J >= columnas)
    nop

Jcorrected:
    mult I, columnas    // I * filas
    mflo idx
    add idx, idx, J     // + J

    add addr, matriz, idx   //matriz[I * columnas + J]
```

```
    lbu val, 0(addr)

    addu v0, v0, val    // v+= matriz[I * columnas + J]

continue:
    add x, x, one       // x++

    ble x, one, loopX   // x <= 1
    nop

    add y, y, one       // y++

    ble y, one, loopY   // y <= 1
    nop

    lw gp, 4(sp)        // Restaurar gp y fp
    lw fp, 0(sp)        // desdde la SRA
    addu sp, sp, 8      // Destruir el stack
    jr ra               //return v

IltZero:
    add I, I, filas     //I += filas
    b Icorrected
    nop

IeqFilas:
    move I, zero        //I = 0
    b Icorrected
    nop

JltZero:
    add J, J, columnas  //J += columnas
    b Jcorrected
    nop

JeqColumnas:
    move J, zero    //J = 0
    b Jcorrected
    nop

.end vecinos
```

vecinos.S

6.0.5. Makefile

```
CFLAGS = -O0 -Wall -Werror -pedantic -pedantic-errors
```

```
SRC := src
SOURCES := $(SRC)/main.c $(SRC)/vecinos.c

machine := $(shell uname -m)
ifndef C_IMPL
    ifeq ($(machine),mips64)
        SOURCES := $(SRC)/main.c $(SRC)/vecinos.S
    endif
endif

all: conway

conway:
    $(CC) $(SOURCES) -o conway -lm $(CFLAGS)

clean:
    $(RM) -f conway *.pbm
```

Makefile