66.20 Organización de Computadoras

Facultad de Ingeniería, Universidad de Buenos Aires

Trabajo práctico 1: Conjunto de instrucciones MIPS

Padrón	Alumno	Dirección de correo
100680	Javier Ferreyra	jferreyra@fi.uba.ar
100490	Julián Crespo	jcrespo@fi.uba.ar
99139	Fernando Sinisi	fsinisi@fi.uba.ar

Primer Cuatrimestre de 2020

Práctica Jueves

Índice

1. Introducción

Se realizó una versión en lenguaje C del “Juego de la Vida”. El programa tiene 2 versiones, una realizada completamente en C y otra que tiene una función principal en C desde la cual se llama una subrutina programada en código MIPS32. Se utiliza el emulador QEMU para simular el entorno de desarrollo, con una máquina MIPS que corre una versión de Debian.

2. Documentación relevante al diseño e implementación del programa

2.1. Comandos

Se cuenta con diferentes comandos que facilitan el uso del programa:

Mostrar la versión del programa

```
$ conway -V
```

Mostrar la ayuda

```
$ conway -h
```

Ejecutar el programa

```
$ conway <iteraciones> <filas> <columnas> <archivo_entrada> [-o <prefijo_salida>]
```

Por ejemplo, si queremos realizar 5 iteraciones, con una matriz de 10x10, archivo de entrada `glider` y que los archivos de salida sean de la forma `estado_nnn.pbm`, debemos introducir el siguiente comando.

```
$ conway 5 10 10 glider -o estado
```

2.2. Diseño

El diseño del programa consiste en un programa principal y un TDA matriz. El programa principal se encarga de realizar todas las comprobaciones necesarias de los parámetros de entrada y de los archivos, y la matriz se encarga de guardar el estado del tablero, modificarlo (incluyendo realizar una iteración del juego) e imprimirlo a un archivo.

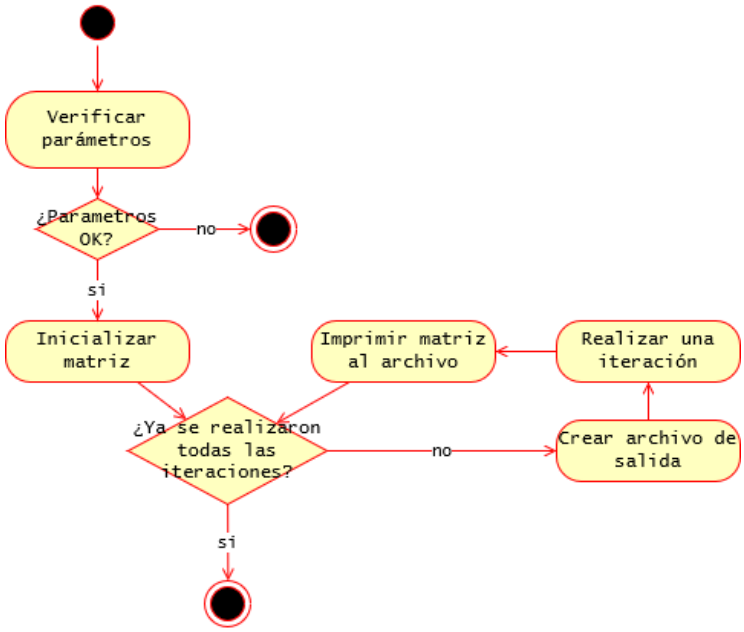


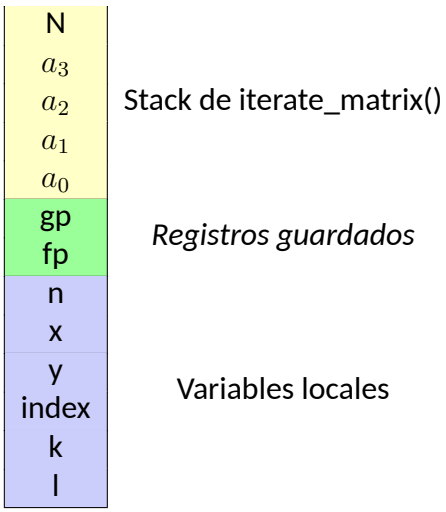
Figura 1: Diagrama de flujo general del programa

2.3. Detalles de implementación

2.3.1. vecinos.S

En este caso la función que se solicitaba que esté implementada en código Assembly MIPS32 fue incluida en el programa como una función auxiliar del TDA Matriz, el cual la utilizaba en su método *iterate_matrix()*. Este método es el encargado de realizar una iteración del juego recorriendo todos los casilleros y calculando los vecinos prendidos de cada uno, para eso utilizaba la función *vecinos()* escrita en Assembly.

La función fue implementada como una función hoja, por lo tanto no se requería guardar en el stack el registro *$ra*. Lo que sí resultó necesario fue crear un área de variables locales ya que no fue posible implementar la función utilizando únicamente registros temporales. El stack resultante fue el siguiente:



Las variables que se indican (x, y, l, , etc) son análogas y cumplen el mismo propósito que las utilizadas en la versión de la función en C.

3. Comandos para compilar el programa

El proyecto cuenta con un Makefile para facilitar el proceso de compilación en las diferentes versiones del programa, ya sea con la función vecinos en C o en Assembly, así como también para crear un video y limpiar los archivos de salida.

Compilar el programa en sus versiones C o Assembly MIPS:

```
$ make conway
$ make conway-mips
```

Nota: se debe estar en el guest corriendo debian sobre mips para compilar la versión de Assembly

Crear un video (el parámetro *name* puede omitir en cuyo caso se utilizará *out*)

```
$ make video name=<nombre>
```

Nota: El script utilizará todos los archivos .pbm que encuentre en la carpeta actual.

Limpiar archivos de salida:

```
$ make clean
```

Compilar con un compilador diferente

```
$ make conway cc=clang
```

Agregar flags de compilación

```
$ make conway flags='-ggdb'
```

Cambiar ruta del ejecutable

```
$ make conway path='/home'
```

Mostrar la ayuda

```
$ make help
```

4. Corrida de Pruebas

Las pruebas realizadas se basaron en los ejemplos del enunciado. Se probaron los comandos básicos como -h y -V para probar que muestren el resultado esperado. Luego, se realizaron pruebas con diferentes archivos de entrada utilizando 10 iteraciones y una matriz de 20x20:

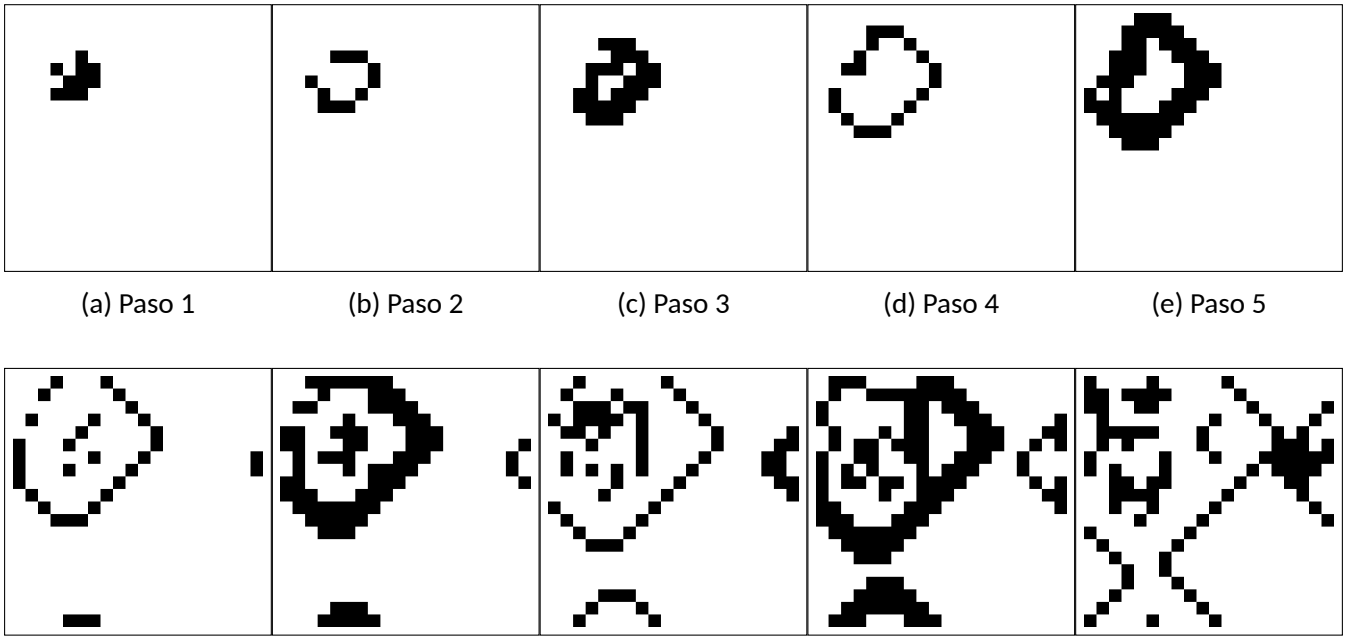
4.1. Salida al ingresar parámetros incorrectos

```
javier@Javier-PC:~/Facultad/Orga de Compus/Repo/TP1$ bin/conway 5 1 0 glider
Error: Parametros incorrectos. Pruebe conway -h para ver ejemplo
javier@Javier-PC:~/Facultad/Orga de Compus/Repo/TP1$ bin/conway 5 1 1 glider
Ejecutando juego...

La matriz no tiene el tamaño adecuado
Error, Ejecucion fallida de convay
javier@Javier-PC:~/Facultad/Orga de Compus/Repo/TP1$ bin/conway 5 1 1 asd
Ejecutando juego...

Error, no se puede abrir el archivo
javier@Javier-PC:~/Facultad/Orga de Compus/Repo/TP1$ bin/conway
Error: Numero de parametros incorrecto.
javier@Javier-PC:~/Facultad/Orga de Compus/Repo/TP1$ bin/conway 1 2 3
Error: Numero de parametros incorrecto.
javier@Javier-PC:~/Facultad/Orga de Compus/Repo/TP1$ bin/conway 5 10 10 glider -o
Error: Numero de parametros incorrecto.
javier@Javier-PC:~/Facultad/Orga de Compus/Repo/TP1$ bin/conway 5 10 10 glider -o estado a a
Error: Numero de parametros incorrecto.
```

4.2. glider

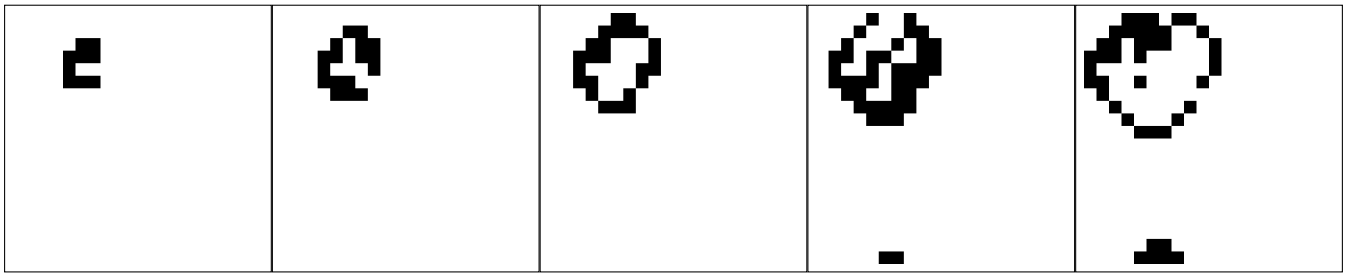


(a) Paso 1 (b) Paso 2 (c) Paso 3 (d) Paso 4 (e) Paso 5

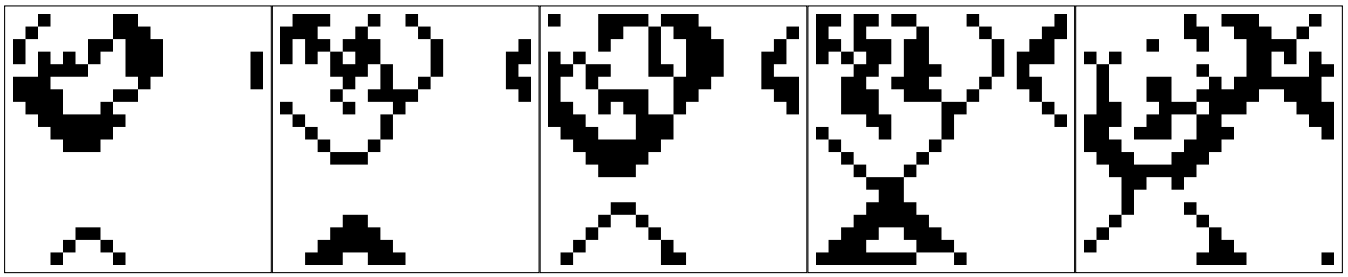
(f) Paso 6 (g) Paso 7 (h) Paso 8 (i) Paso 9 (j) Paso 10

Figura 2: Evolución de las células indicadas en el archivo *glider*

4.3. pento



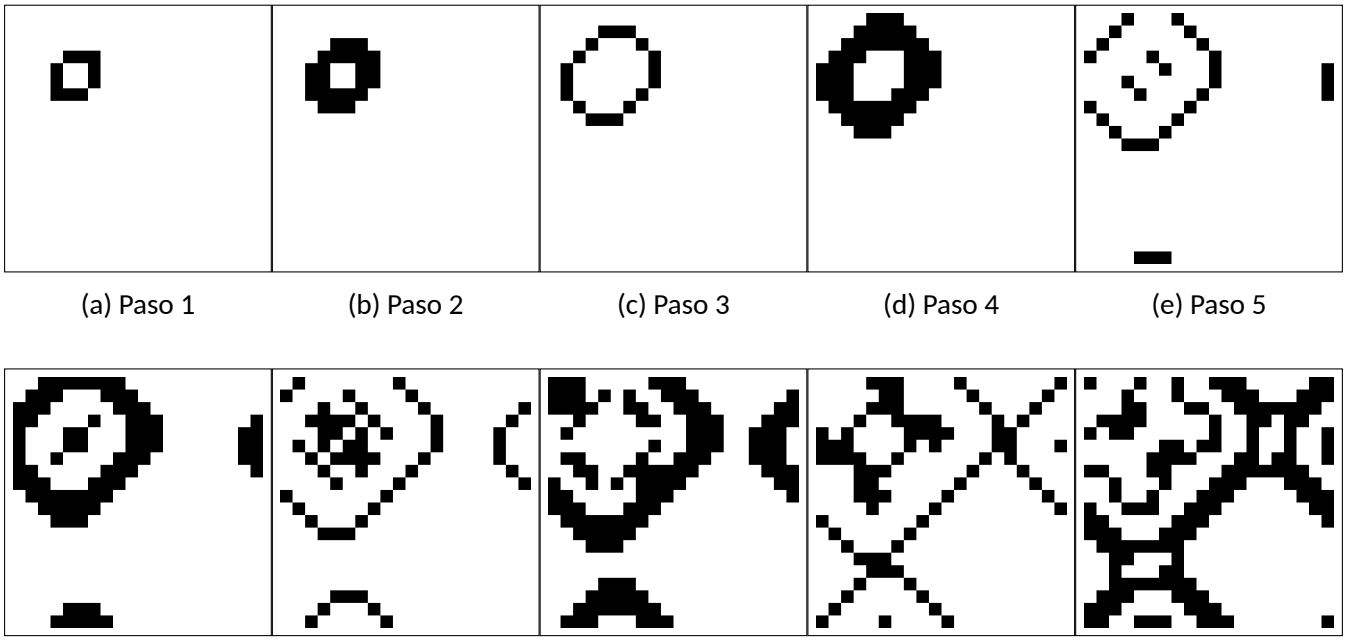
(a) Paso 1 (b) Paso 2 (c) Paso 3 (d) Paso 4 (e) Paso 5



(f) Paso 6 (g) Paso 7 (h) Paso 8 (i) Paso 9 (j) Paso 10

Figura 3: Evolución de las células indicadas en el archivo *pento*

4.4. sapo



(a) Paso 1 (b) Paso 2 (c) Paso 3 (d) Paso 4 (e) Paso 5

(f) Paso 6 (g) Paso 7 (h) Paso 8 (i) Paso 9 (j) Paso 10

Figura 4: Evolución de las células indicadas en el archivo *sapo*

4.5. Valgrind

Al tratarse de un programa que utiliza memoria dinámica, debemos asegurarnos de que no haya fugas de memoria, para eso utilizamos Valgrind, a continuación se muestra el resultado de la corrida de la herramienta con el ejecutable del presente trabajo en una máquina Ubuntu.

```
==12956== Memcheck, a memory error detector
==12956== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==12956== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==12956== Command: bin/conway 5 10 10 glider
==12956==
Ejecutando juego...

Grabando glider_001.pbm...OK
Grabando glider_002.pbm...OK
Grabando glider_003.pbm...OK
Grabando glider_004.pbm...OK
Grabando glider_005.pbm...OK
==12956==
==12956== HEAP SUMMARY:
```

```
==12956==     in use at exit: 0 bytes in 0 blocks
==12956==   total heap usage: 21 allocs, 21 frees, 29,648 bytes allocated
==12956==
==12956== All heap blocks were freed -- no leaks are possible
==12956==
==12956== For counts of detected and suppressed errors, rerun with: -v
==12956== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

5. Conclusiones

En conclusión, se cumplió el objetivo del Trabajo Práctico ya que se desarrolló el programa detallado por el enunciado. Los comandos del programa se ejecutan como detalla el comando -h y tras ser sometidos a distintas pruebas se concluye que funcionan según lo esperado. Así mismo fue posible la implementación del código en MIPS32 y se pudo utilizar el QEMU para simular un entorno de desarrollo de una máquina MIPS corriendo el sistema operativo Debian para ejecutarlo.

6. Anexo

6.1. Código fuente en C

6.1.1. conway.c

```c
#define _POSIX_C_SOURCE 200809L

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#include "matrix.h"

#define OPT_HELP "-h"
#define OPT_VERSION "-V"
#define OUT_PREFIX "-o"

const char VERSION[] = "Version 1.0\n";

const char HELP[] =
"Uso:\n\
\tconway -h\n\
\tconway -V\n\
\tconway i M N inputfile [-o outputprefix]\nOpciones:\n\
\t-h, --help Imprime este mensaje.\n\
\t-V, --versión Da la versión del programa.\n\
```

```
\t-o Prefijo de los archivos de salida.\n\
Ejemplos:\n\
\tconway 10 20 20 glider -o estado\n\
Los archivos de salida se llamarán estado_n.pbm.\n\
Si no se da un prefijo para los archivos de salida,\n\
el prefijo será el nombre del archivo de entrada.\n";

/* ********************************************************************
 *                      FUNCIONES AUXILIARES
 * ********************************************************************/

//convierte string a numero
unsigned int get_num(char* str){
    if(str[0]=='0' && strlen(str)==1) return 0;
    for(int i = 0; i < strlen(str)-1;i++){
        if(!isdigit(str[i]))return -1;
    }
    int num = atoi(str);
    if (num==0) return -1;
    return num;
}

//verifica los parámetros de entrada del programa
int verify_argv(int argc, char *argv[]) {

    if (argc == 2) {
        if (!strcmp(argv[1], OPT_HELP)) {
            fprintf(stdout, HELP);
            return 1;
        }
        if (!strcmp(argv[1], OPT_VERSION)) {
            fprintf(stdout, VERSION);
            return 1;
        }
        fprintf(stderr, "Parametro desconocido: %s\n", argv[1]);
        return -1;
    }
    if (argc == 5 || argc == 7) {
        int i = get_num(argv[1]);
        int m = get_num(argv[2]);
        int n = get_num(argv[3]);

        int numbers_ok =  i > 0 && m > 0 && n > 0;
        int prefix_needed = argc == 7;

        if ((prefix_needed && !strcmp(argv[5], OUT_PREFIX)) || !prefix_needed){
            if (numbers_ok) {
                fprintf(stdout, "%s\n", "Ejecutando juego...\n");
                return 0;
            }
        }
```

```
        fprintf(stderr, "%s\n",
            "Error: Parametros incorrectos. Pruebe conway -h para ver ejemplo");
        return -1;
    }

    fprintf(stderr, "%s\n", "Error: Numero de parametros incorrecto.");
    return -1;
}

//agrega una linea del archivo a la matriz
int add_values(char* line,size_t len, matrix_t* matrix_a ){
    const char *space = " ";
    char *token;
    int num;
    int cont = 0;
    unsigned int pos[2];
    int r;

    token = strtok(line, space);

    while( token != NULL ) {
        num = get_num(token);
        if(num == -1 || cont >= 2){
            fprintf(stderr, "%s\n", "Error: Archivo de entrada invalido");
            return -1;
        }
        pos[cont] = num;
        cont++;
        token = strtok(NULL, space);
    }
    if(cont != 2){
        fprintf(stderr, "%s\n", "Error: Archivo de entrada invalido");
        return -1;
    }
    r = add_value(matrix_a, pos);
    if (r){
        fprintf(stderr, "%s\n", "La matriz no tiene el tamaño adecuado");
        return 1;
    }
    return 0;
}

//lee el archivo y agrega los valores a la matriz usando add_values
int init_matrix(matrix_t* matrix_a,FILE * input_file){

    int r;
    char* line = NULL;
    size_t size = 0;

    size_t len = getline(&line, &size, input_file);
    while(len != -1){
```

```
        r = add_values(line,len,matrix_a);
        if(r != 0){
            free((void*)line);
            return -1;;
        }
        len = getline(&line, &size, input_file);
    }
    free((void*)line);
    return 0;
}

/* ***************************************************************
 *                    FUNCIONES PRINCIPALES
 * ***************************************************************/

//realiza el loop principal del juego
int run(unsigned int i, unsigned int m, unsigned int n ,
        FILE * input_file, char* output_prefix){

    int r;
    matrix_t* matrix_a = create_matrix(m, n);
    if (!matrix_a) {
        return -2;
    }

    r = init_matrix(matrix_a,input_file);
    if(r != 0){
        destroy_matrix(matrix_a);
        return -1;
    }

    size_t prefix_len = strlen(output_prefix);
    for (int j = 0; j < i; j++){
        /*
        Obtengo el nombre del archivo de salida:
        [output_prefix]_[j].pbm
        */
        char filename[50] = {0};
        char num[4] = {0};
        sprintf(num, "%03d", j+1);
        strcpy(filename, output_prefix);
        strcpy(&filename[prefix_len], "_");
        strcpy(&filename[prefix_len+1], num);
        strcpy(&filename[prefix_len+1+3], ".pbm");

        //Creo el archivo
        FILE* output = fopen(filename, "w+");

        //Hago la iteracion del juego
        r = iterate_matrix(matrix_a);
        if (r) return -1;
```

```
        //Imprimo matriz
        fprintf(stdout, "Grabando %s...", filename);
        fflush(stdout);
        r = print_matrix(output, matrix_a);
        if (r) return -1;
        fprintf(stdout, "OK\n");

        //Guardo y cierro archivo
        fflush(output);
        fclose(output);
    }

    destroy_matrix(matrix_a);
    return 0;
}

/*
 * realiza las comprobaciones iniciales y lanza el juego usando
 * las funciones anteriores
 */
int main(int argc, char *argv[]) {

    int r = verify_argv(argc, argv);
    if (r > 0) {
        return 0;
    } else if (r < 0) {
        return -r;
    }

    FILE *input_file = fopen(argv[4],"r");

    if(!input_file){
        fprintf(stderr, "Error, no se puede abrir el archivo\n");
        return -1;
    }

    unsigned int i = get_num(argv[1]);
    unsigned int m = get_num(argv[2]);
    unsigned int n = get_num(argv[3]);

    if (argc == 7){
        r = run(i, m, n, input_file, argv[6]);
    } else {
        r = run(i, m, n, input_file, argv[4]);
    }

    if (r < 0) {
        fprintf(stderr, "Error, Ejecucion fallida de convay\n");
        exit(-r);
    }
```

```
    fclose(input_file);
    return 0;
}
```

6.1.2. matrix.h

```
#ifndef MATRIX_H
#define MATRIX_H

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct matrix {
    unsigned int rows;
    unsigned int cols;
    unsigned char* array;
};

typedef struct matrix matrix_t;

// Constructor de matrix_t
matrix_t* create_matrix(unsigned int rows, unsigned int cols);

// Destructor de matrix_t
void destroy_matrix(matrix_t* m);

// Imprime matrix_t sobre el file pointer fp en el formato solicitado por el enunciado
int print_matrix(FILE* fp, matrix_t* m);

//Agrega un valor 1 (celda viva) en la coordenada indicada
int add_value(matrix_t* matrix_a, unsigned int* pos);

//Realiza una iteracion del juego
int iterate_matrix(matrix_t* matrix);


#endif
```

6.1.3. matrix.c

```
#include <errno.h>
#include "matrix.h"
#define DOT_SIZE 64

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

```
void check_fprint(FILE* fp, int copy);

unsigned int coordToArrayIndex(matrix_t* matrix, int x, int y){
    return x*matrix->cols + y;
}

matrix_t* create_matrix(unsigned int rows, unsigned int cols) {

    matrix_t* matrix = malloc(sizeof(matrix_t));
    if (!matrix) return NULL;

    matrix->rows = rows;
    matrix->cols = cols;

    unsigned int n = rows*cols;
    matrix->array = malloc(sizeof(unsigned char)* n);
    memset(matrix->array, 0, sizeof(unsigned char) * n);
    if (!matrix->array) {
        free(matrix);
        return NULL;
    }

    return matrix;
}

void destroy_matrix(matrix_t* m) {
    if (m->array){
        free(m->array);
        m->array = NULL;
    }
    if (m){
        free(m);
        m = NULL;
    }
}

int add_value(matrix_t* matrix_a, unsigned int *pos){

    int index = coordToArrayIndex(matrix_a, pos[0], pos[1]);
    int limit = matrix_a->rows * matrix_a->cols;
    if (index >= limit){
        return 1;
    }

    matrix_a->array[index] = 1;
    return 0;
}


int print_matrix(FILE* fp, matrix_t* m) {
    int copy;
```

```
    /*El formato de archivo .pbm es:

    *    P1
    *    M N           (tamaño de la matriz)
    *     1 1 1 0 0  (cada numero 1 es un px "pintado")
    *     0 0 0 1 0
    *     0 1 1 0 1
    */

    copy = fprintf(fp,"P1\n");
    check_fprint(fp, copy);
    copy = fprintf(fp, "%d", m->rows*DOT_SIZE);
    check_fprint(fp, copy);
    copy = fprintf(fp, " ");
    copy = fprintf(fp, "%d", m->cols*DOT_SIZE);
    check_fprint(fp, copy);
    copy = fprintf(fp, "\n");
    check_fprint(fp, copy);
    for (int i = 0; i < m->rows; i++){
        for (int k = 0; k < DOT_SIZE; k++){
            for (int j = 0; j < m->cols; j++){
                int index = coordToArrayIndex(m, i, j);
                for (int l = 0; l < DOT_SIZE; l++){
                    copy = fprintf(fp, "%d ", m->array[index]);
                    check_fprint(fp, copy);
                }
            }
            copy = fprintf(fp, "\n");
            check_fprint(fp, copy);
        }
    }
    fflush(fp);
    return 0;
}

void check_fprint(FILE* fp, int copy) {
    if (copy < 0) {
        fprintf(stderr,"Error en la copia del archivo resultante\n");
        exit(EXIT_FAILURE);
    }
}

int iterate_matrix(matrix_t* matrix) {

    unsigned char* next_matrix = malloc(sizeof(unsigned char)
                                        *matrix->rows
                                        *matrix->cols);
    if (!next_matrix)
        return 1;
```

```
    for (unsigned int row = 0; row < matrix->rows; row++){
        for (unsigned int col = 0; col < matrix->cols; col++){
            unsigned int n = vecinos(matrix->array, row, col,
                                     matrix->rows, matrix->cols);
            int index = coordToArrayIndex(matrix, row, col);
            if ((n < 2) || (n > 3)){
                next_matrix[index] = 0;
            } else {
                next_matrix[index] = 1;
            }
        }
    }

    if (matrix->array)
        free(matrix->array);
    matrix->array = next_matrix;

    return 0;
}
```

6.1.4. vecinos.c

```
#include <stdio.h>

unsigned int vecinos(unsigned char *a, unsigned int i, unsigned int j,
                     unsigned int M, unsigned int N){
    unsigned int n, x, y, index;
    int k, l;
    n = 0;
    for (k = -1; k <= 1; k++){
        for (l = -1; l <= 1; l++){
            if ((k == 0) && (l == 0))
                continue;
            x = (M+i+k) % M;
            y = (N+j+l) % N;
            /*
            * Explicacion: uso (M+i+k) y (N+j+l) en vez de solo (i+k)
            * y (j+l) porque la operacion modulo de C funciona mal con
            * numeros negativos (por ejemplo cuando tengo la posicion
            * 0,0 y quiero revisar a la izquierda me queda -1,0). Por
            * eso le sumo M y N respectivamente asi no es negativo y el
            * modulo da lo mismo.
            */
            index = N*x+y;
            if (a[index] == 1)
                n++;
        }
    }
    return n;
}
```

6.2. Código fuente en MIPS32

6.2.1. vecinos.S

```
#include <regdef.h>

#define SS 32
#define O_N 48
#define O_M 44
#define O_j 40
#define O_i 36
#define O_a 32
#define O_GP 28
#define O_FP 24
#define O_n 20
#define O_x 16
#define O_y 12
#define O_index 8
#define O_k 4
#define O_l 0

        .data
        .align 2

        .text
        .align 2
        .globl vecinos
        .ent vecinos

vecinos:
        .frame  fp, SS, ra
        .set    noreorder
        .cpload t9
        .set    reorder
        subu sp, sp, SS

        sw   gp, O_GP(sp)    # salvo gp
        sw   fp, O_FP(sp)    # salvo fp
        move fp, sp

        sw   a0, O_a(fp)
        sw   a1, O_i(fp)
        sw   a2, O_j(fp)
        sw   a3, O_M(fp)

        sw zero, O_n(fp)


        li t1, -2           # t1 es el registro que usaremos de auxiliar para
                            # la carga de constantes
        sw t1, O_k(fp)      # guardo el -2 en el stack frame posicion 4
                            # (k = iterador del primer for)
```

```
        sw t1, 0_l(fp)      # guardo el -2 en el stack frame posicion 0
                            # (l = iterador del segundo for)
        li t1, 1            # t1 = 1 -> es la condicion de corte de ambos for
                            # for k<=1 o l<=1

for1:
        lw t2, 0_k(fp)      # cargo en t2 el valor actual de k
        addi t2, t2, 1      # aumento el iterador t2 en 1
        bgt t2, t1, done    # si el iterador t2 es mayor a 1 (t1), salto a
                            # etiqueta done
        sw t2, 0_k(fp)      # guardo el nuevo valor de k (aumentado) en el stack
                            # frame en posicion 4
        li t1, -2           # aux = -2
        sw t1, 0_l(fp)      # al terminar un ciclo del primer for vuelvo a poner
                            # en el valor inicial al iterador del segundo for

for2:
        lw t3, 0_l(fp)      # cargo en t2 el valor actual de l
        li t1, 1            # t1 = 1
        addi t3, t3, 1      # aumento el iterador t3 en 1
        bgt t3, t1, for1    # si el iterador t3 mayor a 1 (t1), salto a etiqueta for1
        sw t3, 0_l(fp)      # guardo el nuevo valor de l (aumentado) en el stack
                            # frame en posicion 0

        or t1, t3, t2       # k o l = t1
        beqz t1, for2       # si el resultado de k y l es 0 salto a la rtiqueta for2
                            # (continue)

        add t1, a3, a1      # sumo M + i
        add t1, t1, t2      # sumo (M+i) + k
        rem t1, t1, a3      # resto de (M+i+k) / M = x
        sw t1, 0_x(fp)      # guardo el valor de x (t1) en el stack frame posicion 16

        lw t4, 0_N(fp)      # recupero el valor de N en t4
        add t1, t4, a2      # sumo N + j
        add t1, t1, t3      # sumo (N+j) + l
        rem t1, t1, t4      # resto de (N+j+l) / N
        sw t1, 0_y(fp)      # guardo el valor de y (t1) en el stack frame posicion 12

        lw t1, 0_x(fp)      # recupero el valor de x en t1
        mul t4, t4, t1      # multiplico  N * x
        lw t1, 0_y(fp)      # recupero el valor de y en t1
        add t4, t4, t1      # sumo (x*N) + y (t4 = index)

        addu t5, a0, t4     # a la direccion de comienzo del array (a0) le sumo index
                            # (unsigned char = 1 byte)
        lb t5, 0(t5)        # t5 es el byte del array en la posicion index
        li t1, 1            # cargo 1 en t1 para comparar contra la posicion actual del
                            # array
        bne t5, t1, for2    # salto a la etiqueta for si el array[index] no contiene 1
```

```
        lw t1, 0_n(fp)      # recupero el valor de n actual
        addi t1, t1, 1      # aumento el valor de n
        sw t1, 0_n(fp)      # guardo valor de n actual en el stack frame en la
                            # posicion 20
        b for2              # salto a la etiqueta for2

done:
        lw v0, 0_n(fp)      # obtengo el valor actual de n y lo guardo en el registro de
                            #  retorno v0

        lw fp, 0_FP(sp)     # Destruimos el frame.
        addu    sp, sp, SS

        # Retorno
        j ra

        .end vecinos
        .align 2
```

6.3. Makefile

```
RM_FLAGS := -f -v

CFLAGS := $(flags)
CFLAGS += -Wall -g -O0
MIPSFLAGS := $(flags)
MIPSFLAGS += -Wall -mips32 -mlong32 -g -O0

ifndef $(cc)
    CC := gcc
else
    CC := $(cc)
endif

ifndef name
    name = out
endif

ifndef path
    path := bin
endif

conway: *.c
    $(CC) $(CFLAGS) -o $(path)/$@ $^

conway-mips: conway.c matrix.c vecinos.S
    $(CC) $(MIPSFLAGS) -o $(path)/$@ $^

help:
    @cat help.txt
```

```
video:
    @ffmpeg -f image2 -r 1 -pattern_type glob -i '*.pbm' $(name).avi

.PHONY clean:
    @rm $(RM_FLAGS) *.pbm
    @rm $(RM_FLAGS) $(path)/*
    @rm $(RM_FLAGS) *.avi
```