
UBA FACULTAD DE INGENIERÍA

66.20 Organización de Computadoras

Trabajo Practico

2^{do} Cuatrimestre 2016

Grupo 1

Integrantes:

Federico Rodriguez	93336
fedelonghi@hotmail.com	
Ezequiel Dufau	91985
fedelonghi@hotmail.com	
Pablo Ascarza	89711
fedelonghi@hotmail.com	

Índice

1. Enunciado	2
2. Introducción	2
3. Utilización	2
3.1. Compilación y Ejecución	2
3.2. Documentación de Parámetros	2
3.3. Documentación de Errores	3
3.4. Algunas Aclaraciones	3
3.5. Ejemplos de Uso	3
4. Implementación	4
4.1. Implementación en C	4
4.2. Implementación en MIPS	5
4.2.1. Diagrama del Stack de Vecinos	5
5. Corridas de Prueba	5
6. Conclusiones	7
A. Código Fuente	7

1. Enunciado

2. Introducción

El presente trabajo tiene como objetivo familiarizarse con el conjunto de instrucciones *MIPS-32* y el concepto de ABI. Para el cumplimiento de este objetivo se desarrolló un programa que simula el “*Juego de la Vida*” de *Conway* según lo detallado en el enunciado.

La implementación se realizó en el lenguaje de programación C. Además se desarrolló una porción en assembler *MIPS-32* que luego será detallada.

El programa fue desarrollado para correr sobre una plataforma *NetBSD* / *MIPS-32* mediante el emulador *GXEmul*.

3. Utilización

El programa fue implementado para que cumpliera con los requisitos pedidos por el tp. En las siguientes secciones se detallarán los diferentes aspectos para la ejecución del programa.

3.1. Compilación y Ejecución

1. Descargar el archivo fuente “conway.c”
2. Compilar el archivo (por ejemplo con gcc:

```
gcc -Wall -c ‘conway.c’  
gcc -Wall -o ‘conway’ ‘conway.c’
```

)
3. Ejecutar el programa con: `./conway i M N input [-o output]`

3.2. Documentación de Parámetros

- `i` es la cantidad de simulaciones que queremos realizar.
- `M` y `N` especifican las dimensiones de la matriz sobre la cual queremos simular.
- `input` es el nombre del archivo que contiene las coordenadas de las celdas vivas e identifica el estado inicial de la matriz.
- `-o` es un parámetro opcional que especifica que se utilizará el nombre `output` como prefijo de los nombres de los archivos pbm generados. En caso de no existir este parámetro tomara como prefijo `input`.
- `-V` o `--version` muestra la versión del programa.
- `-h` o `--help` muestra la ayuda.

3.3. Documentación de Errores

A continuación se detallan los errores y su significado:

- **Actions Count must be a positive integer:** El primer parámetro tiene que ser un entero positivo.
- **Number of rows must be a positive integer:** El número de filas tiene que ser un entero positivo.
- **Number of columns must be a positive integer:** El número de columnas tiene que ser un entero positivo.
- **Invalid parameter:** Para el caso de `-V`, `-h` y `-o`. El parámetro no coincide con estos valores (o sus equivalentes).
- **Wrong number of parameters:** Hay parámetros de mas o de menos (se pasó un número de parámetros distinto a 1 o 6).
- **Error while opening input file:** No se pudo abrir el archivo de entrada.
- **Error while opening output file: [nombre_de_archivo]:** No se pudo abrir el archivo de salida con el nombre *nombre_de_archivo*

3.4. Algunas Aclaraciones

- Las imágenes pbm generadas se guardan en la carpeta imágenes.
- Todos los errores se imprimen directamente a `stderr`.

3.5. Ejemplos de Uso

Para ver la documentación:

```
./conway -h
```

Para ver la informacion sobre la version:

```
./conway -V
```

Para generar un tablero de 100×50 a partir del archivo `glider` y realizar 20 iteraciones:

```
./conway 20 100 50 glider
```

Los archivos pbm generados por el comando anterior seran nombrados de la forma: “glider_N.pbm”.

Para generar un tablero de 20×30 a partir del archivo **pento**, realizar 10 iteraciones y que los archivos pbm generados tengan como prefijo el nombre **jorge**:

```
./conway 10 20 30 pento -o jorge
```

Los archivos pbm generados por el comando anterior seran nombrados de la forma: “jorge_N.pbm”.

4. Implementación

En esta sección se presentará porciones del programa. Para ver el código completo dirigirse al apéndice A.

La implementación se hizo completamente en C. Luego se programó en MIPS la función **vecinos**. A continuación se detallan las dos implementaciones.

4.1. Implementación en C

Para la implementación se diseñó una función en C según la documentación del enunciado que corresponde a:

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

Tenemos que considerar un detalle sobre el tratamiento de la matriz para entender el algoritmo. En la figura siguiente se muestra la conversión de la matriz a array que utilizamos.

$$\begin{bmatrix} 0,0 & 0,1 & 0,2 \\ 1,0 & 1,1 & 1,2 \\ 2,0 & 2,1 & 2,2 \end{bmatrix} \rightarrow \begin{bmatrix} 0,0 & 0,1 & 0,2 & 1,0 & 1,1 & 1,2 & 2,0 & 2,1 & 2,2 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

A continuación se muestra el algoritmo implementado en C:

Código 1: Código de la función **vecinos**

```
1 unsigned int vecinos(unsigned char *a, unsigned int i, unsigned  
2   int j, unsigned int M, unsigned int N){  
3   unsigned int vecinos = 0;  
4   int x,y;  
5   for (x = -1; x<2; x++){  
       for (y = -1; y<2; y++){
```

```

6         if (x+i>=0 && x+i<N && y+j>=0 && y+j<M){
7             if ((!(x==0&&y==0)) && a[N*(x+i) + (y+j)] == '1'){
8                 vecinos++;
9             }
10        }
11    }
12 }
13 return vecinos;
14 }

```

4.2. Implementación en MIPS

4.2.1. Diagrama del Stack de Vecinos

5. Corridas de Prueba

Realizamos varias corridas de prueba con los archivos proporcionados (pento, glider y sapo). Mostramos a continuación algunas salidas de las corridas:

```
./conway 10 10 10 glider -s
```

```

1 fedelonghi@fedelonghi-dell $ ./conway 5 10 10 glider -s
2 starting action
3 Simulation number: 0
4
5 .....
6 .....
7 .....
8 ....0.....
9 .....0....
10 ...000....
11 .....
12 .....
13 .....
14 .....
15 Simulation number: 1
16
17 .....
18 .....
19 .....
20 .....
21 ...0.0....
22 ....00....
23 ....0.....
24 .....
25 .....
26 .....
27 Simulation number: 2
28
29 .....
30 .....
31 .....

```

```

32 .....
33 .....0....
34 ...0.0....
35 ....00....
36 .....
37 .....
38 .....
39 Simulation number: 3
40 .....
41 .....
42 .....
43 .....
44 .....
45 ....0....
46 .....00...
47 ....00....
48 .....
49 .....
50 .....
51 Simulation number: 4
52 .....
53 .....
54 .....
55 .....
56 .....
57 .....0....
58 .....0....
59 ....000...
60 .....
61 .....
62 .....

```

```
./test 10 10 10 glider -o archivo_salida
```

```

1 fedelonghi@fedelonghi-dell $ ./test 10 10 10 glider -o
  archivo_salida
2 starting action

```

Los archivos .pbm fueron generados en el directorio imagenes.

```
./test 3 10 10 sapo -s
```

```

1 fedelonghi@fedelonghi-dell $ ./test 3 10 10 sapo -s
2 starting action
3 Simulation number: 0
4 .....
5 .....
6 .....
7 .....
8 .....
9 ....000...
10 ...000....
11 .....
12 .....
13 .....

```

```

14 .....
15 Simulation number: 1
16
17 .....
18 .....
19 .....
20 .....0....
21 ...0..0...
22 ...0..0...
23 ....0.....
24 .....
25 .....
26 .....
27 Simulation number: 2
28
29 .....
30 .....
31 .....
32 .....
33 ....000...
34 ...000....
35 .....
36 .....
37 .....
38 .....

```

6. Conclusiones

Apéndice

A. Código Fuente

Código 2: cornway.c

```

1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4  #include <stdbool.h>
5
6  void writePBM(unsigned char** board, unsigned int dimx,
7               unsigned int dimy, const char* fileName, unsigned int
8               actionNumber)
9  {
10     int i, j;
11     int actionNumberStringLength = 1;
12     if (actionNumber >= 10){
13         actionNumberStringLength = 2;
14     }
15     char str[actionNumberStringLength];

```



```

14 sprintf(str, "%d", actionNumber);
15 char newFileName[50];
16 newFileName[0]='\0';
17 strcat(newFileName, "imagenes/");
18 strcat(newFileName, fileName);
19 strcat(newFileName, "_");
20 strcat(newFileName, str);
21 strcat(newFileName, ".pbm");
22
23 FILE *fp = fopen(newFileName, "wb");
24 if (fp==NULL){
25     fprintf(stderr, "Error while opening output file: %s\n",
        newFileName);
26     exit(1);
27 }
28
29 unsigned int amp = 1;
30 (void) fprintf(fp, "P4\n%d %d\n", dimy*amp, dimx*amp);
31 for (j = 0; j < dimy; ++j){
32     for (i = 0; i < dimx; ++i)
33     {
34         unsigned int x,y;
35         unsigned char writeValue = 0;
36         if ((board[i][j]) == '1'){
37             writeValue = 1;
38         }
39         for (x = i*amp; x < amp*(i+1); x++){
40             for (y = j*amp; y < amp*(j+1); y++){
41                 fprintf(fp, "%c", writeValue);
42             }
43         }
44     }
45 }
46 (void) fclose(fp);
47 return;
48 }
49
50
51 unsigned char ** init_board(unsigned int rows, unsigned int
    cols){
52     unsigned char** board = (unsigned char**)malloc(cols*sizeof(
        char*));
53     unsigned int i,j;
54     for (i = 0; i<cols; i++){
55         board[i] = (unsigned char *) malloc(rows*sizeof(char*));
56         for (j = 0; j<rows; j++){
57             board[i][j]='\0';
58         }
59     }
60     return board;
61 }
62
63

```

```

64 void load_board(unsigned char **board, unsigned int rows,
    unsigned int cols, FILE *file){
65     int x,y;
66     while (fscanf (file, "%i %i", &x, &y)!=EOF){
67         if (x<cols||y<rows){
68             board[x][y]='1';
69         }
70     }
71 }
72
73
74 int free_board (unsigned char **board, unsigned int rows,
    unsigned int cols){
75     unsigned int i;
76     for (i = 0; i<cols; i++){
77         free(board[i]);
78     }
79     free(board);
80     return 0;
81 }
82
83
84 void print_board(unsigned char **board, unsigned int rows,
    unsigned int cols){
85     unsigned int i,j;
86     for (i = 0; i<cols; i++){
87         for (j = 0; j<rows; j++){
88             if (board[i][j] == '0'){
89                 printf(".");
90             }else{
91                 printf("0");
92             }
93         }
94         printf("\n");
95     }
96 }
97
98 void print_board_file(unsigned char **board, unsigned int rows,
    unsigned int cols,int n,const char *filename){
99     char newFilename [50];
100     int numLength = 1;
101     if (n >= 10 && n <= 99){
102         numLength = 2;
103     }else if (n >= 100 && n <= 999){
104         numLength = 3;
105     }
106     char str[numLength];
107     sprintf(str, "%d", n);
108     newFilename[0] = '\0';
109     strcat(newFilename, "output/");
110     strcat(newFilename, filename);
111     strcat(newFilename, "_");
112     strcat(newFilename, str);
113     FILE *file = fopen(newFilename,"w+");

```

```

114 fprintf(file, "Simulacion N: %i\n",n);
115 unsigned int i,j;
116 for (i = 0; i<cols; i++){
117     for (j = 0; j<rows; j++){
118         if (board[i][j] == '0'){
119             fprintf(file, ".");
120         }else{
121             fprintf(file, "0");
122         }
123     }
124     fprintf(file, "\n");
125 }
126 fclose(file);
127 }
128
129
130 unsigned char* board_to_array(unsigned char ** board, unsigned
    int rows, unsigned int cols){
131     unsigned char* array = (unsigned char*)malloc(sizeof(char)*
        rows*cols);
132     int x,y;
133     for (x=0; x<cols; x++){
134         for (y=0; y<rows; y++){
135             unsigned int pos = cols*x+y;
136             array[pos]=board[x][y];
137         }
138     }
139     return array;
140 }
141
142
143 unsigned int vecinos(unsigned char *a, unsigned int i, unsigned
    int j, unsigned int M, unsigned int N){
144     unsigned int vecinos = 0;
145     int x,y;
146     for (x = -1; x<2; x++){
147         for (y = -1; y<2; y++){
148             if (x+i>=0 && x+i<N && y+j>=0 && y+j<M){
149                 if ((!(x==0&&y==0)) && a[N*(x+i) + (y+j)] == '1'){
150                     vecinos++;
151                 }
152             }
153         }
154     }
155     return vecinos;
156 }
157
158
159 unsigned int vecinos_m(unsigned char **board, unsigned int x,
    unsigned int y, unsigned int rows, unsigned int cols){
160     unsigned int vecinos = 0;
161     int i,j;
162     for (i = -1; i<2; i++){
163         for (j = -1; j<2; j++){

```

```

164 if (x+i>=0 && x+i<cols && y+j>=0 && y+j<rows){
165 if ((!(i==0&&j==0)) && board[x+i][y+j] == '1'){
166 vecinos++;
167 }
168 }
169 }
170 }
171 return vecinos;
172 }
173
174
175 void copy_board(unsigned char **from, unsigned char **to,
    unsigned int rows, unsigned int cols){
176 unsigned int i,j;
177 for (i = 0; i<cols; i++){
178 for (j = 0; j<rows; j++){
179 to[i][j] = from[i][j];
180 }
181 }
182 }
183
184
185 void process_board(unsigned char **board, unsigned int rows,
    unsigned int cols, unsigned int actionCount, const char*
    fileName, bool screen){
186 printf ("starting action\n");
187 unsigned char **thisBoard = init_board(rows, cols);
188 copy_board(board, thisBoard, rows, cols);
189 unsigned int i,x,y;
190 for (i = 0; i < actionCount; i++){
191 unsigned char **nextBoard = init_board(rows, cols);
192 if (screen){
193 printf("Simulation number: %i\n\n", i);
194 print_board(thisBoard, rows, cols);
195 print_board_file(thisBoard, rows, cols, i, fileName);
196 }else{
197 writePBM(thisBoard, rows, cols, fileName, i);
198 }
199 for (x = 0; x < cols; x++){
200 for (y = 0; y < rows; y++){
201 unsigned char * array = board_to_array(thisBoard, rows, cols);
202 unsigned int neighbours = vecinos(array, x, y, rows, cols);
203 free(array);
204 //unsigned int neighbours = vecinos_m(thisBoard, x, y, rows, cols);
205 if (thisBoard[x][y]=='1'){
206 if (neighbours == 3 || neighbours == 2){nextBoard[x][y] = '1';}
207 }else{
208 if (neighbours == 3){nextBoard[x][y] = '1';}
209 }
210 }
211 }
212 copy_board(nextBoard, thisBoard, rows, cols);
213 free_board(nextBoard, rows, cols);
214 }

```

```

215 free_board(thisBoard, rows, cols);
216 }
217
218
219 void print_help(){
220 printf("Uso:\n");
221 printf("  conway -h\n  conway -V\n  conway i M N inputfile [-o\n
    outputprefix]\n");
222 printf("Opciones:\n");
223 printf("  -h, --help      Imprime este mensaje\n");
224 printf("  -V, --version   Da la version del programa\n");
225 printf("  -o              Prefijo de los archivos de salida\n");
226 printf("Ejemplos: \n");
227 printf("  conway 10 20 20 glider -o estado\n");
228 printf("  Representa 10 iteraciones del Juego de la Vida en una\n
    Matriz\n");
229 printf("  de 20x20, con un estado inicial tomado del archivo ''\n
    glider''.\n");
230 printf("  Los archivos de salida se llamarán estado_n.pbm,\n");
231 printf("  si no se da un prefijo para el archivo de salida, \n"
    );
232 printf("  el prefijo será el nombre del archivo de entrada.\n")
    ;
233 }
234
235
236 void print_version (){
237 printf("Conway -Game of Life- version: 1.0\n");
238 }
239
240
241 void validate_actionsCount(int actionsCount){
242 if (actionsCount <= 0){
243 fprintf(stderr, "Actions Count must be a positive integer!\n");
244 exit(1);
245 }
246 }
247
248
249 void validate_rows(int rows){
250 if (rows <= 0){
251 fprintf(stderr, "Number of rows must be a positive integer!\n")
    ;
252 exit(1);
253 }
254 }
255
256
257 void validate_cols(int cols){
258 if (cols <= 0){
259 fprintf(stderr, "Number of columns must be a positive integer!\n"
    n");
260 exit(1);
261 }

```

```

262 }
263
264
265 int main(int argc, char* argv[])
266 {
267     char const *fileName;
268     char const *outputFileName;
269     unsigned int actionsCount;
270     unsigned int rows;
271     unsigned int cols;
272     unsigned char **board;
273     bool screen = 0;
274     //asigno valores de parametros
275     if (argc == 2){
276         char* arg = argv[1];
277         if (strcmp(arg, "-h") == 0 || strcmp(arg, "--help") == 0){
278             print_help();
279             return 0;
280         }
281         else if (strcmp(arg, "-v") == 0 || strcmp(arg, "--version") == 0)
282         {
283             print_version();
284             return 0;
285         }
286         else{
287             fprintf(stderr, "Invalid parameter!\n");
288             exit(1);
289         }
290     }else if (argc != 5 && argc != 7 && argc != 6){ //ver
291         fprintf(stderr, "Wrong number of parameters!\n");
292         exit(1);
293     }else{
294         fileName = argv[4];
295         actionsCount = (int) atoi(argv[1]);
296         validate_actionsCount(actionsCount);
297         rows = (int) atoi(argv[2]);
298         validate_rows(rows);
299         cols = (int) atoi(argv[3]);
300         validate_cols(cols);
301         outputFileName = fileName;
302         if (argc == 7){
303             if (strcmp(argv[5], "-o") == 0){
304                 outputFileName = argv[6];
305             }else{
306                 fprintf(stderr, "Invalid parameter!\n");
307                 exit(1);
308             }
309         }
310         if (argc == 6){
311             if (strcmp(argv[5], "-s") == 0){
312                 screen = 1;
313             }else{
314                 fprintf(stderr, "Invalid parameter!\n");
315                 exit(1);

```

```
315 }
316 }
317
318 }
319
320 FILE* file = fopen(fileName, "r");
321 if (file==NULL){
322     fprintf(stderr, "Error while opening input file\n");
323     exit(1);
324 }
325
326 board = init_board(rows, cols);
327 load_board(board, rows, cols, file);
328 fclose(file);
329 process_board(board, rows, cols, actionsCount, outputFileName,
330               screen);
331 free_board(board, rows, cols);
332
333 return 0;
334 }
```