

Trabajo Práctico N°0
“Infraestructura básica”

Belén Beltran, Padrón Nro. 91.718
belubeltran@gmail.com

Pablo Ariel Rodriguez, Padrón Nro. 93.970
prodriguez@fi.uba.ar

Federico Martín Rossi, Padrón Nro. 92.086
federicomrossi@gmail.com

2do. Cuatrimestre 2013
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

Índice

| | |
|---|-----------|
| 1. Introducción | 1 |
| 2. Compilación | 1 |
| 3. Utilización | 1 |
| 4. Implementación | 1 |
| 4.1. Implementación en C | 1 |
| 4.1.1. Algoritmo <i>Pgm</i> | 2 |
| 4.2. Generación de código del algoritmo <i>Pgm</i> en Assembly MIPS32 | 3 |
| 5. Debugging | 6 |
| 6. Pruebas | 6 |
| 6.1. Pruebas al ingreso y egreso de datos del programa | 6 |
| 6.2. Tiempos de ejecución | 8 |
| Appendices | 10 |
| A. Implementación completa en lenguaje C | 10 |
| A.1. <i>tp0.c</i> . Implementación del main del programa | 10 |
| A.2. <i>pgm.h</i> . Declaración de la librería Pgm | 11 |
| A.3. <i>pgm.c</i> . Definición de la librería Pgm | 11 |
| B. Generación del código completo en assembly MIPS | 12 |
| B.1. <i>tp0.s</i> . Generación del main del programa | 12 |
| B.2. <i>pgm.s</i> . Generación del algoritmo Pgm | 16 |

1. Introducción

En el presente trabajo se tiene como objetivo la familiarización con las herramientas de software que se utilizarán a lo largo de los siguientes trabajos prácticos, llevando a cabo la implementación de un programa que resuelve cierta problemática piloto detallada en los próximos apartados.

La implementación se realizará en el lenguaje de programación C. Luego se ejecutará la aplicación sobre una plataforma *NetBSD/MIPS-32* mediante el emulador *GXEmul* [1]. Finalmente generaremos, mediante el compilador *GCC* [2], el equivalente en assembler MIPS del código fuente en C.

Todos los archivos y códigos fuente aquí mencionados, así como también el presente informe, pueden ser descargados como un archivo comprimido ZIP del repositorio del grupo¹.

2. Compilación

La herramienta para compilar el código en lenguaje C será el *GCC*.

Para automatizar las tareas de compilación se hace uso de la herramienta *GNU Make*. Los Makefiles utilizados para la compilación se incluyen junto al resto de los archivos fuentes del presente trabajo.

3. Utilización

Veamos ahora la forma en la que debe ser ejecutado el programa implementado en lenguaje C. El resultado de la compilación con “make” será un programa ejecutable, de nombre *tp0*, que podrá ser invocado con los siguientes parámetros:

- *-h*: Imprime ayuda para la utilización del programa;
- *-V*: Imprimir la versión actual del programa;
- *-r [Width]x[Height]*: Setea la resolución en píxeles del bitmap;
- *-o [Path]*: Especifica la ruta del archivo de salida sobre el cual se guarda el tablero generado por la aplicación.

Entonces, si se desea generar un tablero con una resolución de 100x100 píxeles sobre el archivo de nombre *imagen.pgm*, debe ejecutarse el comando de la siguiente manera:

```
$ ./tp0 -r 100x100 -o imagen.pgm
```

En caso de no desear generar un archivo, y solo se espera ver el contenido del archivo por salida estándar, debe ejecutarse el comando de la siguiente manera:

```
$ ./tp0 -r 100x100 -o -
```

4. Implementación

En lo que sigue de la sección, se presentarán los códigos fuente de la implementación del algoritmo. Aquellos lectores interesados en la implementación completa del programa, pueden dirigirse al apéndice ubicado al final del presente informe.

4.1. Implementación en C

La implementación del programa fue dividida en los siguientes módulos:

- **tp0**: Programa principal responsable de interpretar los parámetros especificados a través de la terminal de modo de que realice las tareas solicitadas por el usuario. Su función es, de acuerdo al parámetro o los parámetros especificados, llevar a cabo la ejecución solicitada haciendo uso de módulos externos;

¹URI del Repositorio: <https://github.com/federicomrossi/6620-trabajos-practicos-2C2013/tree/master/tp0>

- **pgm**: Módulo encargado de generar una imagen de un tablero de ajedrez (cuadrado, de ocho casilleros de lado) conforme a los parámetros ingresados. Se utiliza para ello el formato gráfico PGM o *portable gray map*. De especificarse un archivo de salida, la imagen en formato PGM se almacenará en este medio.

4.1.1. Algoritmo *Pgm*

En el *Código 1* se muestra el header de la librería, donde se declara la función *pgm()*, mientras que en el *Código 2* se muestra la definición de la librería.

Código 1: “pgm.h”

```

1 #ifndef PGP_H_INCLUDED
2 #define PGP_H_INCLUDED
3 #include <stdio.h>
4
5 /**
6  * x: ancho, y: alto, fileName: Nombre del archivo a grabar
7  */
8 void pgm(unsigned x,unsigned y, char* fileName);
9
10 #endif // PGP_H_INCLUDED

```

Código 2: “pgm.c”

```

1 #include "pgm.h"
2 #include <string.h>
3
4 void escribirLinea(unsigned int x, FILE* file,char start){
5     unsigned int i;
6     unsigned int change = x >> 3;
7     unsigned char resto = x & 0x00000007;
8     unsigned int count = 0;
9     for(i=0;i<x;i++){
10         fprintf(file,"%d ",start);
11         count++;
12         if(count == change + (resto ? 1 : 0)){
13             start = !start;
14             if(resto){
15                 resto--;
16             }
17             count=0;
18         }
19     }
20     fprintf(file,"\n");
21 }
22
23 void pgm(unsigned x,unsigned y, char* fileName){
24     FILE* file;
25     int i;
26     char start=1;
27     unsigned int change = y >> 3;
28     unsigned char resto = y & 0x00000007;
29     unsigned int count = 0;
30     if(!strcmp(fileName,"-")){
31         file = stdout;
32     }else{
33         file = fopen(fileName,"w");
34     }
35     fprintf(file,"P2\n# %s\n%d %d\n1\n", fileName, x, y);
36     for(i=0;i<y;i++){
37         escribirLinea(x,file,start);
38         count++;
39         if(count == change + (resto ? 1 : 0)){
40             start = !start;
41             if(resto)
42                 resto--;
43             count=0;
44         }
45     }

```

```

46     if(file != stdout)
47         fclose(file);
48 }

```

4.2. Generación de código del algoritmo *Pgm* en Assembly MIPS32

A continuación se presenta el código en Assembly MIPS32 del algoritmo *Pgm* generado automáticamente por el compilador *GCC*. En el *Apéndice A* se encuentra el código del programa en su totalidad.

Código 3: “*pgm.s*”

```

1  .file 1 "pgm.c"
2  .section .mdebug.abi32
3  .previous
4  .abicalls
5  .rdata
6  .align 2
7  $LC0:
8  .ascii "%d \000"
9  .align 2
10 $LC1:
11 .ascii "\n\000"
12 .text
13 .align 2
14 .globl escribirLinea
15 .ent escribirLinea
16 escribirLinea:
17 .frame $fp,64,$31 # vars= 24, regs= 3/0, args= 16, extra= 8
18 .mask 0xd0000000,-8
19 .fmask 0x00000000,0
20 .set noreorder
21 .cpload $25
22 .set reorder
23 subu $sp,$sp,64
24 .cprestore 16
25 sw $31,56($sp)
26 sw $fp,52($sp)
27 sw $28,48($sp)
28 move $fp,$sp
29 sw $4,64($fp)
30 sw $5,68($fp)
31 move $2,$6
32 sb $2,24($fp)
33 lw $2,64($fp)
34 srl $2,$2,3
35 sw $2,32($fp)
36 lbu $2,64($fp)
37 andi $2,$2,0x7
38 sb $2,36($fp)
39 sw $0,40($fp)
40 sw $0,28($fp)
41 $L6:
42 lw $2,28($fp)
43 lw $3,64($fp)
44 sltu $2,$2,$3
45 bne $2,$0,$L9
46 b $L7
47 $L9:
48 lb $2,24($fp)
49 lw $4,68($fp)
50 la $5,$LC0
51 move $6,$2
52 la $25,fprintf
53 jal $31,$25
54 lw $2,40($fp)
55 addu $2,$2,1
56 sw $2,40($fp)
57 lbu $2,36($fp)
58 beq $2,$0,$L11
59 lw $2,32($fp)

```

```

60  addu $3,$2,1
61  lw $2,40($fp)
62  beq $2,$3,$L12
63  b $L8
64  $L11:
65  lw $3,40($fp)
66  lw $2,32($fp)
67  beq $3,$2,$L12
68  b $L8
69  $L12:
70  lb $2,24($fp)
71  xori $2,$2,0x0
72  sltu $2,$2,1
73  sb $2,24($fp)
74  lbu $2,36($fp)
75  beq $2,$0,$L13
76  lbu $2,36($fp)
77  addu $2,$2,-1
78  sb $2,36($fp)
79  $L13:
80  sw $0,40($fp)
81  $L8:
82  lw $2,28($fp)
83  addu $2,$2,1
84  sw $2,28($fp)
85  b $L6
86  $L7:
87  lw $4,68($fp)
88  la $5,$LC1
89  la $25,fprintf
90  jal $31,$25
91  move $sp,$fp
92  lw $31,56($sp)
93  lw $fp,52($sp)
94  addu $sp,$sp,64
95  j $31
96  .end escribirLinea
97  .size escribirLinea, .-escribirLinea
98  .rdata
99  .align 2
100 $LC2:
101 .ascii "-\000"
102 .align 2
103 $LC3:
104 .ascii "w\000"
105 .align 2
106 $LC4:
107 .ascii "P2\n"
108 .ascii "# %s\n"
109 .ascii "%d %d\n"
110 .ascii "1\n\000"
111 .text
112 .align 2
113 .globl pgm
114 .ent pgm
115 pgm:
116 .frame $fp,72,$31 # vars= 24, regs= 3/0, args= 24, extra= 8
117 .mask 0xd0000000,-8
118 .fmask 0x00000000,0
119 .set noreorder
120 .cplod $25
121 .set reorder
122 subu $sp,$sp,72
123 .cpstore 24
124 sw $31,64($sp)
125 sw $fp,60($sp)
126 sw $28,56($sp)
127 move $fp,$sp
128 sw $4,72($fp)
129 sw $5,76($fp)
130 sw $6,80($fp)
131 li $2,1 # 0x1
132 sb $2,40($fp)
133 lw $2,76($fp)
134 srl $2,$2,3
135 sw $2,44($fp)

```



```

136 lbu $2,76($fp)
137 andi $2,$2,0x7
138 sb $2,48($fp)
139 sw $0,52($fp)
140 lw $4,80($fp)
141 la $5,$LC2
142 la $25,strcmp
143 jal $31,$25
144 bne $2,$0,$L15
145 la $2,___sF+88
146 sw $2,32($fp)
147 b $L16
148 $L15:
149 lw $4,80($fp)
150 la $5,$LC3
151 la $25,fopen
152 jal $31,$25
153 sw $2,32($fp)
154 $L16:
155 lw $2,76($fp)
156 sw $2,16($sp)
157 lw $4,32($fp)
158 la $5,$LC4
159 lw $6,80($fp)
160 lw $7,72($fp)
161 la $25,fprintf
162 jal $31,$25
163 sw $0,36($fp)
164 $L17:
165 lw $2,36($fp)
166 lw $3,76($fp)
167 sltu $2,$2,$3
168 bne $2,$0,$L20
169 b $L18
170 $L20:
171 lb $2,40($fp)
172 lw $4,72($fp)
173 lw $5,32($fp)
174 move $6,$2
175 la $25,escribirLinea
176 jal $31,$25
177 lw $2,52($fp)
178 addu $2,$2,1
179 sw $2,52($fp)
180 lbu $2,48($fp)
181 beq $2,$0,$L22
182 lw $2,44($fp)
183 addu $3,$2,1
184 lw $2,52($fp)
185 beq $2,$3,$L23
186 b $L19
187 $L22:
188 lw $3,52($fp)
189 lw $2,44($fp)
190 beq $3,$2,$L23
191 b $L19
192 $L23:
193 lb $2,40($fp)
194 xori $2,$2,0x0
195 sltu $2,$2,1
196 sb $2,40($fp)
197 lbu $2,48($fp)
198 beq $2,$0,$L24
199 lbu $2,48($fp)
200 addu $2,$2,-1
201 sb $2,48($fp)
202 $L24:
203 sw $0,52($fp)
204 $L19:
205 lw $2,36($fp)
206 addu $2,$2,1
207 sw $2,36($fp)
208 b $L17
209 $L18:
210 lw $3,32($fp)
211 la $2,___sF+88

```

```

212 beq $3,$2,$L14
213 lw $4,32($fp)
214 la $25,fclose
215 jal $31,$25
216 $L14:
217 move $sp,$fp
218 lw $31,64($sp)
219 lw $fp,60($sp)
220 addu $sp,$sp,72
221 j $31
222 .end pgm
223 .size pgm, .-pgm
224 .ident "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"

```

5. Debugging

Para analizar el correcto funcionamiento del programa se crearon casos de prueba pertinentes, considerando combinaciones diferentes en el ingreso de parámetros al programa principal, como también tomando en cuenta las diferentes salidas obtenidas a partir del algoritmo *pgm*. Los resultados fueron comparados con los casos esperados y así se determinó el correcto funcionamiento del programa en su totalidad.

6. Pruebas

6.1. Pruebas al ingreso y egreso de datos del programa

A modo de prueba del programa, se ingresaron ciertos parámetros a éste y se analizó la salida obtenida. A continuación se presentan algunas pruebas realizadas y los resultados obtenidos.

El primer caso consiste en crear un tablero de resolución 11x16 y que el resultado sea enviado a la salida estandar. Se debe ingresar por consola lo siguiente:

```
$ ./tp0 -r 11x16 -o -
```

La salida del programa es la siguiente:

```

P2
# -
11 16
1
1 1 0 0 1 1 0 1 0 1 0
1 1 0 0 1 1 0 1 0 1 0
0 0 1 1 0 0 1 0 1 0 1
0 0 1 1 0 0 1 0 1 0 1
1 1 0 0 1 1 0 1 0 1 0
1 1 0 0 1 1 0 1 0 1 0
0 0 1 1 0 0 1 0 1 0 1
0 0 1 1 0 0 1 0 1 0 1
1 1 0 0 1 1 0 1 0 1 0
1 1 0 0 1 1 0 1 0 1 0
0 0 1 1 0 0 1 0 1 0 1
0 0 1 1 0 0 1 0 1 0 1
1 1 0 0 1 1 0 1 0 1 0
1 1 0 0 1 1 0 1 0 1 0
0 0 1 1 0 0 1 0 1 0 1
0 0 1 1 0 0 1 0 1 0 1
1 1 0 0 1 1 0 1 0 1 0
1 1 0 0 1 1 0 1 0 1 0
0 0 1 1 0 0 1 0 1 0 1
0 0 1 1 0 0 1 0 1 0 1

```

La segunda prueba consiste en crear un tablero de resolución 16x11 y también enviar la salida a consola. Se debe ingresar por la línea de comandos:

```
$ ./tp0 -r 16x11 -o -
```

La salida del programa es la siguiente:

```
P2
# -
16 11
1
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
```

La tercer prueba consiste en generar el tablero de tamaño default 8x8 (se genera en caso de no ingresar la opción -r), ingresando por consola:

```
$ ./tp0 -o -
```

El resultado del programa se ve por salida standard y es el siguiente:

```
P2
# -
8 8
1
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
```

Una prueba más interesante, es aquella en la cual la resolución permite ver una imagen bastante amplia. En este caso se considera una resolución de 200x200 y se decide guardar la salida en un archivo. Se debe ingresar por consola:

```
$ ./tp0 -r 200x200 -o 200x200.pgm
```

La salida en formato de imagen *PGM* se muestra en la *Figura 1*.

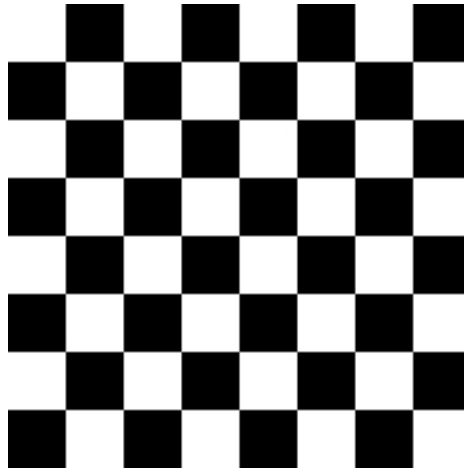


Figura 1: Imágen del tablero de ajedrez con resolución de 200x200 píxeles.

6.2. Tiempos de ejecución

Se quiere medir el tiempo que tarda el programa en la máquina virtual MIPS32 en crear un tablero de ajedrez con ciertas características. Para ello se realizó un módulo aparte que, con la ayuda del comando *GNU "time"* [4], mide los tiempos de la realización de un tablero de resolución 1x1 hasta uno de 1000x1000.

Los datos obtenidos se pueden observar en el *Cuadro 1*.

| Resolución | Tiempo de ejecución [s] |
|------------|-------------------------|
| 1x1 | 0,051 |
| 100x100 | 0,316 |
| 200x200 | 1,148 |
| 300x300 | 2,5 |
| 400x400 | 4,434 |
| 500x500 | 6,844 |
| 600x600 | 9,848 |
| 700x700 | 13,5 |
| 800x800 | 17,449 |
| 900x900 | 22,152 |
| 1000x1000 | 27,613 |

Cuadro 1: *Tiempos en segundos obtenidos en la ejecución del programa para distintas resoluciones.*

En la *Figura 2* se presenta un diagrama resumido de los tiempos de ejecución obtenidos. Como es de esperar, se obtiene una curva cuadrática.

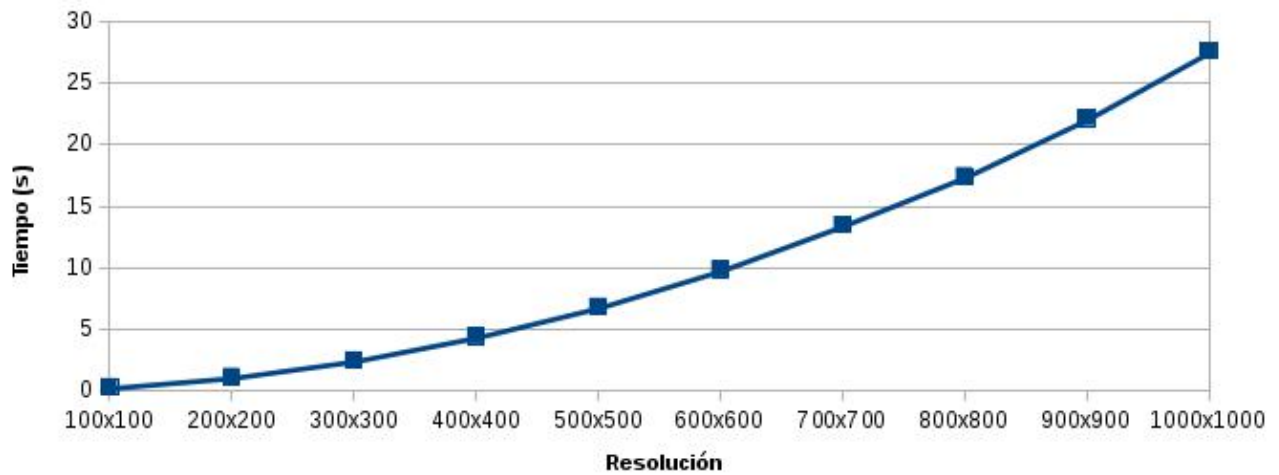


Figura 2: *Gráfico de tiempo de ejecución (en segundos) respecto de la resolución de imagen generada.*

Referencias

- [1] The NetBSD project, <http://www.netbsd.org/>
- [2] GCC, the GNU Compiler Collection, <http://gcc.gnu.org/>
- [3] PGM format specification, <http://netpbm.sourceforge.net/doc/pgm.html>
- [4] time man page, <http://unixhelp.ed.ac.uk/CGI/man-cgi?time>
- [5] J. L. Hennessy and D. A. Patterson, "Computer Architecture. A Quantitative Approach," 4th Edition, Morgan Kaufmann Publishers, 2000.

Apéndices

A. Implementación completa en lenguaje C

A.1. *tp0.c*. Implementación del main del programa

Código 4: “*tp0.c*”

```
1  /*
2  =====
3  Name      : tp0.c
4  Authors   : Belen Beltran (91718)
5             Federico Martin Rossi (92086)
6             Pablo Rodriguez (93970)
7  Version   : 1.0
8  Description : Programa de dibujo de tableros de Ajedrez en formato PGM
9  =====
10 */
11
12 #include <stdio.h>
13 #include <stdlib.h>
14 #include <unistd.h>
15 #include <string.h>
16 #include <stdlib.h>
17 #include "pgm.h"
18
19
20 /** Funciones Auxiliares */
21
22 void displayHelp() {
23     printf("%s", "<- Help display ->\nUsage:\n \
24     tp0 -h\n\
25     tp0 -V\n\
26     tp0 [options]\n\
27     \nOptions:\n\
28     -V, --version Print version and quit.\n\
29     -h, --help Print this information. \n\
30     -r, --resolution Set bitmap resolution to WxH pixels.\n\
31     -o, --output Path to output file.\n\
32     \nExamples:\n\
33     tp0 -o output.pgm\n\
34     tp0 -r 800x600 -o file.pgm \n<- End of help display ->\n");
35 }
36
37 void displayVersion() {
38     printf("%s", "This program draws chess boards in PGM file format\
39     \nVersion: v1.0\n");
40 }
41
42 void getXY(char* optarg, unsigned* x, unsigned* y) {
43     char* occur;
44     if (((occur = strchr(optarg, 'x')) != 0) ||
45         ((occur = strchr(optarg, 'X')) != 0)) {
46         *x = (unsigned)atoi(optarg);
47         *y = (unsigned)atoi(occur + 1);
48     }
49 }
50
51 int main(int argc, char* argv[]) {
52     int opt, finished = 0, i = 1;
53     char* file = NULL;
54     unsigned x = 8, y = 8; // Valores default
55
56     while (((opt = getopt(argc, argv, "hVr:o:")) != -1) && !finished) {
57         // Solo ejecuta opcion si esta es de una sola letra
58         // Ej: ejecuta '-h', pero no '-help'
59         if (argv[i][2] == 0) {
60             i += 2;
61             switch (opt) {
62                 // Help
```

```

63     case 'h': displayHelp();
64         finished = 1;
65         break;
66     // Version
67     case 'V': displayVersion();
68         finished = 1;
69         break;
70     // Resolution
71     case 'r': getXY(optarg, &x, &y);
72         break;
73     // Output
74     case 'o': file = optarg;
75         finished = 1;
76         break;
77     // Ninguna de las anteriores
78     default: finished = 1;
79         printf("Error: Unknown option selected.\n");
80         displayHelp();
81         break;
82 }
83 }
84 else {
85     finished = 1;
86     printf("Error: Unknown option selected.\n");
87     displayHelp();
88 }
89 }
90 if (file)
91     pgm(x, y, file);
92
93 return EXIT_SUCCESS;
94 }

```

A.2. *pgm.h*. Declaración de la librería Pgm

Código 5: “*pgm.h*”

```

1 #ifndef PGP_H_INCLUDED
2 #define PGP_H_INCLUDED
3 #include <stdio.h>
4
5 /**
6  * x: ancho, y: alto, fileName: Nombre del archivo a grabar
7  */
8 void pgm(unsigned x, unsigned y, char* fileName);
9
10 #endif // PGP_H_INCLUDED

```

A.3. *pgm.c*. Definición de la librería Pgm

Código 6: “*pgm.c*”

```

1 #include "pgm.h"
2 #include <string.h>
3
4 void escribirLinea(unsigned int x, FILE* file, char start){
5     unsigned int i;
6     unsigned int change = x >> 3;
7     unsigned char resto = x & 0x00000007;
8     unsigned int count = 0;
9     for(i=0; i<x; i++){
10         fprintf(file, "%d ", start);
11         count++;
12         if(count == change + (resto ? 1 : 0)){

```

```

13         start = !start;
14         if(resto){
15             resto--;
16         }
17         count=0;
18     }
19 }
20 fprintf(file, "\n");
21 }
22
23 void pgm(unsigned x, unsigned y, char* fileName){
24     FILE* file;
25     int i;
26     char start=1;
27     unsigned int change = y >> 3;
28     unsigned char resto = y & 0x00000007;
29     unsigned int count = 0;
30     if(!strcmp(fileName, "-")){
31         file = stdout;
32     }else{
33         file = fopen(fileName, "w");
34     }
35     fprintf(file, "P2\n# %s\n%d %d\n1\n", fileName, x, y);
36     for(i=0; i<y; i++){
37         escribirLinea(x, file, start);
38         count++;
39         if(count == change + (resto ? 1 : 0)){
40             start = !start;
41             if(resto)
42                 resto--;
43             count=0;
44         }
45     }
46     if(file != stdout)
47         fclose(file);
48 }

```

B. Generación del código completo en assembly MIPS

B.1. *tp0.s*. Generación del main del programa

Código 7: "*tp0.s*"

```

1  .file 1 "tp0.c"
2  .section .mdebug.abi32
3  .previous
4  .abicalls
5  .rdata
6  .align 2
7  $LC0:
8  .ascii "%s\000"
9  .align 2
10 $LC1:
11 .ascii "<- Help display ->\n"
12 .ascii "Usage:\n"
13 .ascii " \ttp0 -h\n"
14 .ascii " \ttp0 -V\n"
15 .ascii " \ttp0 [options]\n"
16 .ascii " \t\n"
17 .ascii "Options:\n"
18 .ascii " \t-V, --version Print version and quit.\n"
19 .ascii " \t-h, --help Print this information. \n"
20 .ascii " \t-r, --resolution Set bitmap resolution to WxH pixels.\n"
21 .ascii " \t-o, --output Path to output file.\n"
22 .ascii " \t\n"
23 .ascii "Examples:\n"
24 .ascii " \ttp0 -o output.pgm\n"
25 .ascii " \ttp0 -r 800x600 -o file.pgm \n"
26 .ascii "<- End of help display ->\n\000"

```



```

27 .text
28 .align 2
29 .globl displayHelp
30 .ent displayHelp
31 displayHelp:
32 .frame $fp,40,$31 # vars= 0, regs= 3/0, args= 16, extra= 8
33 .mask 0xd0000000,-8
34 .fmask 0x00000000,0
35 .set noreorder
36 .cload $25
37 .set reorder
38 subu $sp,$sp,40
39 .cprestore 16
40 sw $31,32($sp)
41 sw $fp,28($sp)
42 sw $28,24($sp)
43 move $fp,$sp
44 la $4,$LC0
45 la $5,$LC1
46 la $25,printf
47 jal $31,$25
48 move $sp,$fp
49 lw $31,32($sp)
50 lw $fp,28($sp)
51 addu $sp,$sp,40
52 j $31
53 .end displayHelp
54 .size displayHelp,.-displayHelp
55 .rdata
56 .align 2
57 $LC2:
58 .ascii "This program draws chess boards in PGM file format\t\n"
59 .ascii "Version: v1.0\n\000"
60 .text
61 .align 2
62 .globl displayVersion
63 .ent displayVersion
64 displayVersion:
65 .frame $fp,40,$31 # vars= 0, regs= 3/0, args= 16, extra= 8
66 .mask 0xd0000000,-8
67 .fmask 0x00000000,0
68 .set noreorder
69 .cload $25
70 .set reorder
71 subu $sp,$sp,40
72 .cprestore 16
73 sw $31,32($sp)
74 sw $fp,28($sp)
75 sw $28,24($sp)
76 move $fp,$sp
77 la $4,$LC0
78 la $5,$LC2
79 la $25,printf
80 jal $31,$25
81 move $sp,$fp
82 lw $31,32($sp)
83 lw $fp,28($sp)
84 addu $sp,$sp,40
85 j $31
86 .end displayVersion
87 .size displayVersion,.-displayVersion
88 .align 2
89 .globl getXY
90 .ent getXY
91 getXY:
92 .frame $fp,48,$31 # vars= 8, regs= 4/0, args= 16, extra= 8
93 .mask 0xd0010000,-4
94 .fmask 0x00000000,0
95 .set noreorder
96 .cload $25
97 .set reorder
98 subu $sp,$sp,48
99 .cprestore 16
100 sw $31,44($sp)
101 sw $fp,40($sp)
102 sw $28,36($sp)

```

```

103 sw $16,32($sp)
104 move $fp,$sp
105 sw $4,48($fp)
106 sw $5,52($fp)
107 sw $6,56($fp)
108 lw $4,48($fp)
109 li $5,120 # 0x78
110 la $25, strchr
111 jal $31,$25
112 sw $2,24($fp)
113 lw $2,24($fp)
114 bne $2,$0,$L21
115 lw $4,48($fp)
116 li $5,88 # 0x58
117 la $25, strchr
118 jal $31,$25
119 sw $2,24($fp)
120 lw $2,24($fp)
121 bne $2,$0,$L21
122 b $L19
123 $L21:
124 lw $16,52($fp)
125 lw $4,48($fp)
126 la $25, atoi
127 jal $31,$25
128 sw $2,0($16)
129 lw $16,56($fp)
130 lw $2,24($fp)
131 addu $2,$2,1
132 move $4,$2
133 la $25, atoi
134 jal $31,$25
135 sw $2,0($16)
136 $L19:
137 move $sp,$fp
138 lw $31,44($sp)
139 lw $fp,40($sp)
140 lw $16,32($sp)
141 addu $sp,$sp,48
142 j $31
143 .end getXY
144 .size getXY, .-getXY
145 .rdata
146 .align 2
147 $LC3:
148 .ascii "hVr:o:\000"
149 .align 2
150 $LC4:
151 .ascii "Error: Unknown option selected.\n\000"
152 .text
153 .align 2
154 .globl main
155 .ent main
156 main:
157 .frame $fp,72,$31 # vars= 32, regs= 3/0, args= 16, extra= 8
158 .mask 0xd0000000,-8
159 .fmask 0x00000000,0
160 .set noreorder
161 .cplod $25
162 .set reorder
163 subu $sp,$sp,72
164 .cprestore 16
165 sw $31,64($sp)
166 sw $fp,60($sp)
167 sw $28,56($sp)
168 move $fp,$sp
169 sw $4,72($fp)
170 sw $5,76($fp)
171 sw $0,28($fp)
172 li $2,1 # 0x1
173 sw $2,32($fp)
174 sw $0,36($fp)
175 li $2,8 # 0x8
176 sw $2,40($fp)
177 li $2,8 # 0x8
178 sw $2,44($fp)

```

```

179 $L23:
180 lw $4,72($fp)
181 lw $5,76($fp)
182 la $6,$LC3
183 la $25,getopt
184 jal $31,$25
185 sw $2,24($fp)
186 lw $3,24($fp)
187 li $2,-1 # 0xffffffffffffffff
188 beq $3,$2,$L24
189 lw $2,28($fp)
190 bne $2,$0,$L24
191 lw $2,32($fp)
192 sll $3,$2,2
193 lw $2,76($fp)
194 addu $2,$3,$2
195 lw $2,0($2)
196 addu $2,$2,2
197 lb $2,0($2)
198 bne $2,$0,$L27
199 lw $2,32($fp)
200 addu $2,$2,2
201 sw $2,32($fp)
202 lw $2,24($fp)
203 sw $2,48($fp)
204 li $2,104 # 0x68
205 lw $3,48($fp)
206 beq $3,$2,$L29
207 lw $3,48($fp)
208 slt $2,$3,105
209 beq $2,$0,$L35
210 li $2,86 # 0x56
211 lw $3,48($fp)
212 beq $3,$2,$L30
213 b $L33
214 $L35:
215 li $2,111 # 0x6f
216 lw $3,48($fp)
217 beq $3,$2,$L32
218 li $2,114 # 0x72
219 lw $3,48($fp)
220 beq $3,$2,$L31
221 b $L33
222 $L29:
223 la $25,displayHelp
224 jal $31,$25
225 li $2,1 # 0x1
226 sw $2,28($fp)
227 b $L23
228 $L30:
229 la $25,displayVersion
230 jal $31,$25
231 li $2,1 # 0x1
232 sw $2,28($fp)
233 b $L23
234 $L31:
235 addu $2,$fp,40
236 addu $3,$fp,44
237 lw $4,optarg
238 move $5,$2
239 move $6,$3
240 la $25,getXY
241 jal $31,$25
242 b $L23
243 $L32:
244 lw $2,optarg
245 sw $2,36($fp)
246 li $2,1 # 0x1
247 sw $2,28($fp)
248 b $L23
249 $L33:
250 li $2,1 # 0x1
251 sw $2,28($fp)
252 la $4,$LC4
253 la $25,printf
254 jal $31,$25

```

```

255  la $25,displayHelp
256  jal $31,$25
257  b $L23
258 $L27:
259  li $2,1      # 0x1
260  sw $2,28($fp)
261  la $4,$LC4
262  la $25,printf
263  jal $31,$25
264  la $25,displayHelp
265  jal $31,$25
266  b $L23
267 $L24:
268  lw $2,36($fp)
269  beq $2,$0,$L37
270  lw $4,40($fp)
271  lw $5,44($fp)
272  lw $6,36($fp)
273  la $25,pgm
274  jal $31,$25
275 $L37:
276  move $2,$0
277  move $sp,$fp
278  lw $31,64($sp)
279  lw $fp,60($sp)
280  addu $sp,$sp,72
281  j $31
282 .end main
283 .size main, .-main
284 .ident "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"

```

B.2. *pgm.s*. Generación del algoritmo Pgm

Código 8: “*pgm.s*”

```

1  .file 1 "pgm.c"
2  .section .mdebug.abi32
3  .previous
4  .abicalls
5  .rdata
6  .align 2
7  $LC0:
8  .ascii "%d \000"
9  .align 2
10 $LC1:
11 .ascii "\n\000"
12 .text
13 .align 2
14 .globl escribirLinea
15 .ent escribirLinea
16 escribirLinea:
17 .frame $fp,64,$31      # vars= 24, regs= 3/0, args= 16, extra= 8
18 .mask 0xd0000000,-8
19 .fmask 0x00000000,0
20 .set noreorder
21 .cpload $25
22 .set reorder
23 subu $sp,$sp,64
24 .cpstore 16
25 sw $31,56($sp)
26 sw $fp,52($sp)
27 sw $28,48($sp)
28 move $fp,$sp
29 sw $4,64($fp)
30 sw $5,68($fp)
31 move $2,$6
32 sb $2,24($fp)
33 lw $2,64($fp)
34 srl $2,$2,3
35 sw $2,32($fp)

```

```

36  lbu $2,64($fp)
37  andi $2,$2,0x7
38  sb $2,36($fp)
39  sw $0,40($fp)
40  sw $0,28($fp)
41  $L6:
42  lw $2,28($fp)
43  lw $3,64($fp)
44  sltu $2,$2,$3
45  bne $2,$0,$L9
46  b $L7
47  $L9:
48  lb $2,24($fp)
49  lw $4,68($fp)
50  la $5,$LC0
51  move $6,$2
52  la $25,fprintf
53  jal $31,$25
54  lw $2,40($fp)
55  addu $2,$2,1
56  sw $2,40($fp)
57  lbu $2,36($fp)
58  beq $2,$0,$L11
59  lw $2,32($fp)
60  addu $3,$2,1
61  lw $2,40($fp)
62  beq $2,$3,$L12
63  b $L8
64  $L11:
65  lw $3,40($fp)
66  lw $2,32($fp)
67  beq $3,$2,$L12
68  b $L8
69  $L12:
70  lb $2,24($fp)
71  xori $2,$2,0x0
72  sltu $2,$2,1
73  sb $2,24($fp)
74  lbu $2,36($fp)
75  beq $2,$0,$L13
76  lbu $2,36($fp)
77  addu $2,$2,-1
78  sb $2,36($fp)
79  $L13:
80  sw $0,40($fp)
81  $L8:
82  lw $2,28($fp)
83  addu $2,$2,1
84  sw $2,28($fp)
85  b $L6
86  $L7:
87  lw $4,68($fp)
88  la $5,$LC1
89  la $25,fprintf
90  jal $31,$25
91  move $sp,$fp
92  lw $31,56($sp)
93  lw $fp,52($sp)
94  addu $sp,$sp,64
95  j $31
96  .end  escribirLinea
97  .size escribirLinea, .-escribirLinea
98  .rdata
99  .align 2
100 $LC2:
101 .ascii "-\000"
102 .align 2
103 $LC3:
104 .ascii "w\000"
105 .align 2
106 $LC4:
107 .ascii "P2\n"
108 .ascii "# %s\n"
109 .ascii "%d %d\n"
110 .ascii "1\n\000"
111 .text

```

```

112 .align 2
113 .globl pgm
114 .ent pgm
115 pgm:
116 .frame $fp,72,$31 # vars= 24, regs= 3/0, args= 24, extra= 8
117 .mask 0xd0000000,-8
118 .fmask 0x00000000,0
119 .set noreorder
120 .cpload $25
121 .set reorder
122 subu $sp,$sp,72
123 .cpstore 24
124 sw $31,64($sp)
125 sw $fp,60($sp)
126 sw $28,56($sp)
127 move $fp,$sp
128 sw $4,72($fp)
129 sw $5,76($fp)
130 sw $6,80($fp)
131 li $2,1 # 0x1
132 sb $2,40($fp)
133 lw $2,76($fp)
134 srl $2,$2,3
135 sw $2,44($fp)
136 lbu $2,76($fp)
137 andi $2,$2,0x7
138 sb $2,48($fp)
139 sw $0,52($fp)
140 lw $4,80($fp)
141 la $5,$LC2
142 la $25,strcmp
143 jal $31,$25
144 bne $2,$0,$L15
145 la $2,___sF+88
146 sw $2,32($fp)
147 b $L16
148 $L15:
149 lw $4,80($fp)
150 la $5,$LC3
151 la $25,fopen
152 jal $31,$25
153 sw $2,32($fp)
154 $L16:
155 lw $2,76($fp)
156 sw $2,16($sp)
157 lw $4,32($fp)
158 la $5,$LC4
159 lw $6,80($fp)
160 lw $7,72($fp)
161 la $25,fprintf
162 jal $31,$25
163 sw $0,36($fp)
164 $L17:
165 lw $2,36($fp)
166 lw $3,76($fp)
167 sltu $2,$2,$3
168 bne $2,$0,$L20
169 b $L18
170 $L20:
171 lb $2,40($fp)
172 lw $4,72($fp)
173 lw $5,32($fp)
174 move $6,$2
175 la $25,escribirLinea
176 jal $31,$25
177 lw $2,52($fp)
178 addu $2,$2,1
179 sw $2,52($fp)
180 lbu $2,48($fp)
181 beq $2,$0,$L22
182 lw $2,44($fp)
183 addu $3,$2,1
184 lw $2,52($fp)
185 beq $2,$3,$L23
186 b $L19
187 $L22:

```

```

188  lw  $3,52($fp)
189  lw  $2,44($fp)
190  beq $3,$2,$L23
191  b   $L19
192  $L23:
193  lb  $2,40($fp)
194  xori $2,$2,0x0
195  sltu $2,$2,1
196  sb  $2,40($fp)
197  lbu $2,48($fp)
198  beq $2,$0,$L24
199  lbu $2,48($fp)
200  addu $2,$2,-1
201  sb  $2,48($fp)
202  $L24:
203  sw  $0,52($fp)
204  $L19:
205  lw  $2,36($fp)
206  addu $2,$2,1
207  sw  $2,36($fp)
208  b   $L17
209  $L18:
210  lw  $3,32($fp)
211  la  $2,___sF+88
212  beq $3,$2,$L14
213  lw  $4,32($fp)
214  la  $25,fclose
215  jal $31,$25
216  $L14:
217  move $sp,$fp
218  lw  $31,64($sp)
219  lw  $fp,60($sp)
220  addu $sp,$sp,72
221  j   $31
222  .end  pgm
223  .size pgm, .-pgm
224  .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"

```