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

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 grupo[1].

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*: Imprimer 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 apéndic ubicado al final del presente informe.

4.1. Implementación en C

La implementación del programa fue divida 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;

[1] URI 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"*

```h
#ifndef PGP_H_INCLUDED
#define PGP_H_INCLUDED
#include <stdio.h>

/**
*   x: ancho, y: alto, fileName: Nombre del archivo a grabar
*/
void pgm(unsigned x,unsigned y, char* fileName);

#endif // PGP_H_INCLUDED
```

Código 2: *"pgm.c"*

```c
#include "pgm.h"
#include <string.h>

void escribirLinea(unsigned int x, FILE* file,char start){
    unsigned int i;
    unsigned int change = x >> 3;
    unsigned char resto = x & 0x00000007;
    unsigned int count = 0;
    for(i=0;i<x;i++){
        fprintf(file,"%d ",start);
        count++;
        if(count == change + (resto ? 1 : 0)){
            start = !start;
            if(resto){
                resto--;
            }
            count=0;
        }
    }
    fprintf(file,"\n");
}

void pgm(unsigned x,unsigned y, char* fileName){
    FILE* file;
    int i;
    char start=1;
    unsigned int change = y >> 3;
    unsigned char resto = y & 0x00000007;
    unsigned int count = 0;
    if(!strcmp(fileName,"-")){
        file = stdout;
    }else{
        file = fopen(fileName,"w");
    }
    fprintf(file,"P2\n# %s\n%d %d\n1\n", fileName, x, y);
    for(i=0;i<y;i++){
        escribirLinea(x,file,start);
        count++;
        if(count == change + (resto ? 1 : 0)){
            start = !start;
            if(resto)
                resto--;
            count=0;
        }
    }
```

```
    if(file != stdout)
        fclose(file);
}
```

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"*

```
  .file 1 "pgm.c"
  .section .mdebug.abi32
  .previous
  .abicalls
  .rdata
  .align  2
$LC0:
  .ascii  "%d \000"
  .align  2
$LC1:
  .ascii  "\n\000"
  .text
  .align  2
  .globl  escribirLinea
  .ent  escribirLinea
escribirLinea:
  .frame  $fp,64,$31    # vars= 24, regs= 3/0, args= 16, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,64
  .cprestore 16
  sw  $31,56($sp)
  sw  $fp,52($sp)
  sw  $28,48($sp)
  move  $fp,$sp
  sw  $4,64($fp)
  sw  $5,68($fp)
  move  $2,$6
  sb  $2,24($fp)
  lw  $2,64($fp)
  srl $2,$2,3
  sw  $2,32($fp)
  lbu $2,64($fp)
  andi  $2,$2,0x7
  sb  $2,36($fp)
  sw  $0,40($fp)
  sw  $0,28($fp)
$L6:
  lw  $2,28($fp)
  lw  $3,64($fp)
  sltu  $2,$2,$3
  bne $2,$0,$L9
  b $L7
$L9:
  lb  $2,24($fp)
  lw  $4,68($fp)
  la  $5,$LC0
  move  $6,$2
  la  $25,fprintf
  jal $31,$25
  lw  $2,40($fp)
  addu  $2,$2,1
  sw  $2,40($fp)
  lbu $2,36($fp)
  beq $2,$0,$L11
  lw  $2,32($fp)
```

```
  addu  $3,$2,1
  lw  $2,40($fp)
  beq $2,$3,$L12
  b $L8
$L11:
  lw  $3,40($fp)
  lw  $2,32($fp)
  beq $3,$2,$L12
  b $L8
$L12:
  lb  $2,24($fp)
  xori  $2,$2,0x0
  sltu  $2,$2,1
  sb  $2,24($fp)
  lbu $2,36($fp)
  beq $2,$0,$L13
  lbu $2,36($fp)
  addu  $2,$2,-1
  sb  $2,36($fp)
$L13:
  sw  $0,40($fp)
$L8:
  lw  $2,28($fp)
  addu  $2,$2,1
  sw  $2,28($fp)
  b $L6
$L7:
  lw  $4,68($fp)
  la  $5,$LC1
  la  $25,fprintf
  jal $31,$25
  move  $sp,$fp
  lw  $31,56($sp)
  lw  $fp,52($sp)
  addu  $sp,$sp,64
  j $31
  .end  escribirLinea
  .size escribirLinea, .-escribirLinea
  .rdata
  .align  2
$LC2:
  .ascii  "-\000"
  .align  2
$LC3:
  .ascii  "w\000"
  .align  2
$LC4:
  .ascii  "P2\n"
  .ascii  "# %s\n"
  .ascii  "%d %d\n"
  .ascii  "1\n\000"
  .text
  .align  2
  .globl  pgm
  .ent  pgm
pgm:
  .frame  $fp,72,$31    # vars= 24, regs= 3/0, args= 24, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,72
  .cprestore 24
  sw  $31,64($sp)
  sw  $fp,60($sp)
  sw  $28,56($sp)
  move  $fp,$sp
  sw  $4,72($fp)
  sw  $5,76($fp)
  sw  $6,80($fp)
  li  $2,1      # 0x1
  sb  $2,40($fp)
  lw  $2,76($fp)
  srl $2,$2,3
  sw  $2,44($fp)
```

```
  lbu $2,76($fp)
  andi  $2,$2,0x7
  sb  $2,48($fp)
  sw  $0,52($fp)
  lw  $4,80($fp)
  la  $5,$LC2
  la  $25,strcmp
  jal $31,$25
  bne $2,$0,$L15
  la  $2,__sF+88
  sw  $2,32($fp)
  b $L16
$L15:
  lw  $4,80($fp)
  la  $5,$LC3
  la  $25,fopen
  jal $31,$25
  sw  $2,32($fp)
$L16:
  lw  $2,76($fp)
  sw  $2,16($sp)
  lw  $4,32($fp)
  la  $5,$LC4
  lw  $6,80($fp)
  lw  $7,72($fp)
  la  $25,fprintf
  jal $31,$25
  sw  $0,36($fp)
$L17:
  lw  $2,36($fp)
  lw  $3,76($fp)
  sltu  $2,$2,$3
  bne $2,$0,$L20
  b $L18
$L20:
  lb  $2,40($fp)
  lw  $4,72($fp)
  lw  $5,32($fp)
  move  $6,$2
  la  $25,escribirLinea
  jal $31,$25
  lw  $2,52($fp)
  addu  $2,$2,1
  sw  $2,52($fp)
  lbu $2,48($fp)
  beq $2,$0,$L22
  lw  $2,44($fp)
  addu  $3,$2,1
  lw  $2,52($fp)
  beq $2,$3,$L23
  b $L19
$L22:
  lw  $3,52($fp)
  lw  $2,44($fp)
  beq $3,$2,$L23
  b $L19
$L23:
  lb  $2,40($fp)
  xori  $2,$2,0x0
  sltu  $2,$2,1
  sb  $2,40($fp)
  lbu $2,48($fp)
  beq $2,$0,$L24
  lbu $2,48($fp)
  addu  $2,$2,-1
  sb  $2,48($fp)
$L24:
  sw  $0,52($fp)
$L19:
  lw  $2,36($fp)
  addu  $2,$2,1
  sw  $2,36($fp)
  b $L17
$L18:
  lw  $3,32($fp)
  la  $2,__sF+88
```

```
	beq $3,$2,$L14
	lw	$4,32($fp)
	la	$25,fclose
	jal $31,$25
$L14:
	move	$sp,$fp
	lw	$31,64($sp)
	lw	$fp,60($sp)
	addu	$sp,$sp,72
	j $31
	.end	pgm
	.size pgm, .-pgm
	.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 algorirmo *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
```

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
```

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.*

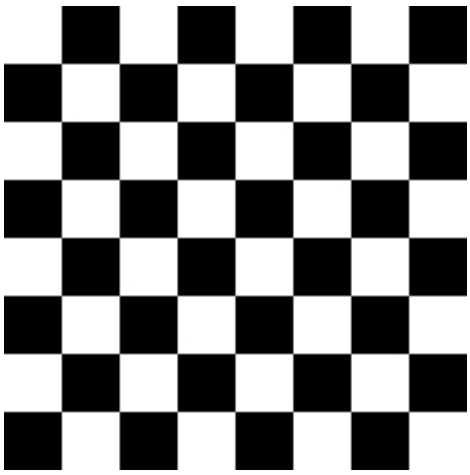

Figura 1: *Imágen del tablero de ajedrez con resolución de 200x200 píxeles.*

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.

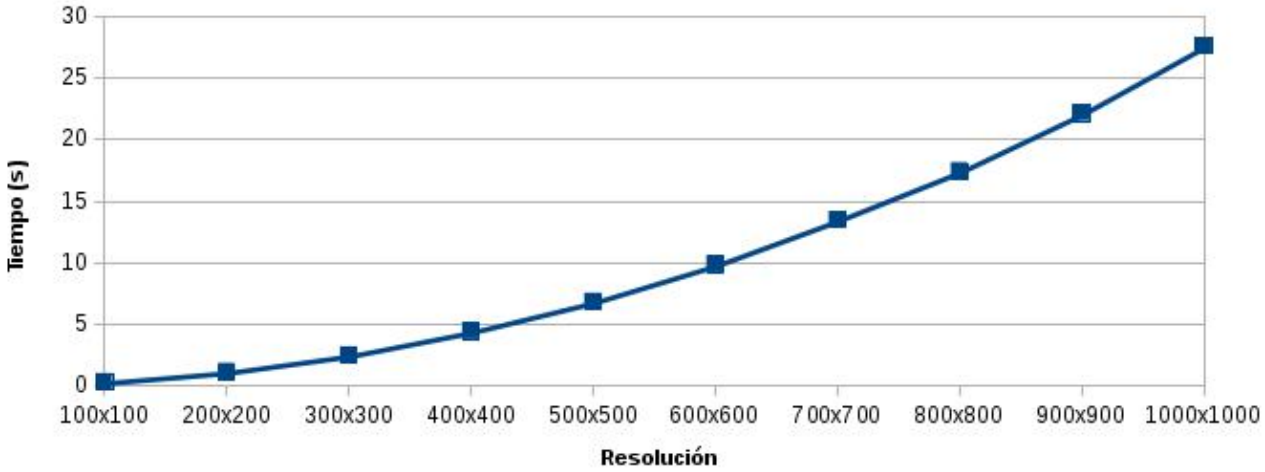

Figura 2: *Gráfico de tiempo de ejecución (en segundos) respecto de la resolución de imagen generada.*

Referencias

[1] The NetBSD project, `http://www.netbsd.org/`

[2] GCC, the GNU Compiler Collection, `http://gcc.gnu.org/`

[3] PGM format speciffication, `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"*

```c
/*
 ============================================================================
 Name        : tp0.c
 Authors     : Belen Beltran (91718)
       Federico Martin Rossi (92086)
       Pablo Rodriguez (93970)
 Version     : 1.0
 Description : Programa de dibujo de tableros de Ajedrez en formato PGM
 ============================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include "pgm.h"


//* Funciones Auxiliares *//

void displayHelp() {
  printf("%s", "<- Help display ->\nUsage:\n \
  tp0 -h\n\
  tp0 -V\n\
  tp0 [options]\n\
  \nOptions:\n\
  -V, --version Print version and quit.\n\
  -h, --help Print this information. \n\
  -r, --resolution Set bitmap resolution to WxH pixels.\n\
  -o, --output Path to output file.\n\
  \nExamples:\n\
  tp0 -o output.pgm\n\
  tp0 -r 800x600 -o file.pgm \n<- End of help display ->\n");
}

void displayVersion() {
  printf("%s", "This program draws chess boards in PGM file format\
  \nVersion: v1.0\n");
}

void getXY(char* optarg, unsigned* x, unsigned* y) {
  char* ocurr;
  if (((ocurr = strchr(optarg, 'x')) != 0) ||
      ((ocurr = strchr(optarg, 'X')) != 0)) {
    *x = (unsigned)atoi(optarg);
    *y = (unsigned)atoi(ocurr + 1);
  }
}

int main(int argc, char* argv[]) {
  int opt, finished = 0, i = 1;
  char* file = NULL;
  unsigned x = 8, y = 8; // Valores default

  while (((opt = getopt(argc, argv, "hVr:o:")) != -1) && !finished) {
    // Solo ejecuta opcion si esta es de una sola letra
    // Ej: ejecuta '-h', pero no '-help'
    if (argv[i][2] == 0) {
      i += 2;
      switch (opt) {
        // Help
```

```
        case 'h':   displayHelp();
              finished = 1;
              break;
        // Version
        case 'V': displayVersion();
              finished = 1;
              break;
        // Resolution
        case 'r': getXY(optarg, &x, &y);
              break;
        // Output
        case 'o':   file = optarg;
              finished = 1;
              break;
        // Ninguna de las anteriores
        default:  finished = 1;
              printf("Error: Unknown option selected.\n");
              displayHelp();
              break;
      }
    }
    else {
      finished = 1;
      printf("Error: Unknown option selected.\n");
      displayHelp();
    }
  }
  if (file)
        pgm(x, y, file);

  return EXIT_SUCCESS;
}
```

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

Código 5: *"pgm.h"*

```
#ifndef PGP_H_INCLUDED
#define PGP_H_INCLUDED
#include <stdio.h>

/**
*   x: ancho, y: alto, fileName: Nombre del archivo a grabar
*/
void pgm(unsigned x,unsigned y, char* fileName);

#endif // PGP_H_INCLUDED
```

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

Código 6: *"pgm.c"*

```
#include "pgm.h"
#include <string.h>

void escribirLinea(unsigned int x, FILE* file,char start){
    unsigned int i;
    unsigned int change = x >> 3;
    unsigned char resto = x & 0x00000007;
    unsigned int count = 0;
    for(i=0;i<x;i++){
        fprintf(file,"%d ",start);
        count++;
        if(count == change + (resto ? 1 : 0)){
```

```
            start = !start;
            if(resto){
                resto--;
            }
            count=0;
        }
    }
    fprintf(file,"\n");
}

void pgm(unsigned x,unsigned y, char* fileName){
    FILE* file;
    int i;
    char start=1;
    unsigned int change = y >> 3;
    unsigned char resto = y & 0x00000007;
    unsigned int count = 0;
    if(!strcmp(fileName,"-")){
        file = stdout;
    }else{
        file = fopen(fileName,"w");
    }
    fprintf(file,"P2\n# %s\n%d %d\n1\n", fileName, x, y);
    for(i=0;i<y;i++){
        escribirLinea(x,file,start);
        count++;
        if(count == change + (resto ? 1 : 0)){
            start = !start;
            if(resto)
                resto--;
            count=0;
        }
    }
    if(file != stdout)
        fclose(file);
}
```

B. Generación del código completo en assembly MIPS

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

Código 7: *"tp0.s"*

```
  .file 1 "tp0.c"
  .section .mdebug.abi32
  .previous
  .abicalls
  .rdata
  .align  2
$LC0:
  .ascii  "%s\000"
  .align  2
$LC1:
  .ascii  "<- Help display ->\n"
  .ascii  "Usage:\n"
  .ascii  " \ttp0 -h\n"
  .ascii  "\ttp0 -V\n"
  .ascii  "\ttp0 [options]\n"
  .ascii  "\t\n"
  .ascii  "Options:\n"
  .ascii  "\t-V, --version Print version and quit.\n"
  .ascii  "\t-h, --help Print this information. \n"
  .ascii  "\t-r, --resolution Set bitmap resolution to WxH pixels.\n"
  .ascii  "\t-o, --output Path to output file.\n"
  .ascii  "\t\n"
  .ascii  "Examples:\n"
  .ascii  "\ttp0 -o output.pgm\n"
  .ascii  "\ttp0 -r 800x600 -o file.pgm \n"
  .ascii  "<- End of help display ->\n\000"
```

```
  .text
  .align  2
  .globl  displayHelp
  .ent  displayHelp
displayHelp:
  .frame  $fp,40,$31    # vars= 0, regs= 3/0, args= 16, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,40
  .cprestore 16
  sw  $31,32($sp)
  sw  $fp,28($sp)
  sw  $28,24($sp)
  move  $fp,$sp
  la  $4,$LC0
  la  $5,$LC1
  la  $25,printf
  jal $31,$25
  move  $sp,$fp
  lw  $31,32($sp)
  lw  $fp,28($sp)
  addu  $sp,$sp,40
  j $31
  .end  displayHelp
  .size displayHelp, .-displayHelp
  .rdata
  .align  2
$LC2:
  .ascii  "This program draws chess boards in PGM file format\t\n"
  .ascii  "Version: v1.0\n\000"
  .text
  .align  2
  .globl  displayVersion
  .ent  displayVersion
displayVersion:
  .frame  $fp,40,$31    # vars= 0, regs= 3/0, args= 16, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,40
  .cprestore 16
  sw  $31,32($sp)
  sw  $fp,28($sp)
  sw  $28,24($sp)
  move  $fp,$sp
  la  $4,$LC0
  la  $5,$LC2
  la  $25,printf
  jal $31,$25
  move  $sp,$fp
  lw  $31,32($sp)
  lw  $fp,28($sp)
  addu  $sp,$sp,40
  j $31
  .end  displayVersion
  .size displayVersion, .-displayVersion
  .align  2
  .globl  getXY
  .ent  getXY
getXY:
  .frame  $fp,48,$31    # vars= 8, regs= 4/0, args= 16, extra= 8
  .mask 0xd0010000,-4
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,48
  .cprestore 16
  sw  $31,44($sp)
  sw  $fp,40($sp)
  sw  $28,36($sp)
```

```
  sw  $16,32($sp)
  move  $fp,$sp
  sw  $4,48($fp)
  sw  $5,52($fp)
  sw  $6,56($fp)
  lw  $4,48($fp)
  li  $5,120      # 0x78
  la  $25,strchr
  jal $31,$25
  sw  $2,24($fp)
  lw  $2,24($fp)
  bne $2,$0,$L21
  lw  $4,48($fp)
  li  $5,88     # 0x58
  la  $25,strchr
  jal $31,$25
  sw  $2,24($fp)
  lw  $2,24($fp)
  bne $2,$0,$L21
  b $L19
$L21:
  lw  $16,52($fp)
  lw  $4,48($fp)
  la  $25,atoi
  jal $31,$25
  sw  $2,0($16)
  lw  $16,56($fp)
  lw  $2,24($fp)
  addu  $2,$2,1
  move  $4,$2
  la  $25,atoi
  jal $31,$25
  sw  $2,0($16)
$L19:
  move  $sp,$fp
  lw  $31,44($sp)
  lw  $fp,40($sp)
  lw  $16,32($sp)
  addu  $sp,$sp,48
  j $31
  .end  getXY
  .size getXY, .-getXY
  .rdata
  .align  2
$LC3:
  .ascii  "hVr:o:\000"
  .align  2
$LC4:
  .ascii  "Error: Unknown option selected.\n\000"
  .text
  .align  2
  .globl  main
  .ent  main
main:
  .frame  $fp,72,$31    # vars= 32, regs= 3/0, args= 16, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,72
  .cprestore 16
  sw  $31,64($sp)
  sw  $fp,60($sp)
  sw  $28,56($sp)
  move  $fp,$sp
  sw  $4,72($fp)
  sw  $5,76($fp)
  sw  $0,28($fp)
  li  $2,1      # 0x1
  sw  $2,32($fp)
  sw  $0,36($fp)
  li  $2,8      # 0x8
  sw  $2,40($fp)
  li  $2,8      # 0x8
  sw  $2,44($fp)
```

```
$L23:
  lw  $4,72($fp)
  lw  $5,76($fp)
  la  $6,$LC3
  la  $25,getopt
  jal $31,$25
  sw  $2,24($fp)
  lw  $3,24($fp)
  li  $2,-1     # 0xffffffffffffffff
  beq $3,$2,$L24
  lw  $2,28($fp)
  bne $2,$0,$L24
  lw  $2,32($fp)
  sll $3,$2,2
  lw  $2,76($fp)
  addu  $2,$3,$2
  lw  $2,0($2)
  addu  $2,$2,2
  lb  $2,0($2)
  bne $2,$0,$L27
  lw  $2,32($fp)
  addu  $2,$2,2
  sw  $2,32($fp)
  lw  $2,24($fp)
  sw  $2,48($fp)
  li  $2,104      # 0x68
  lw  $3,48($fp)
  beq $3,$2,$L29
  lw  $3,48($fp)
  slt $2,$3,105
  beq $2,$0,$L35
  li  $2,86     # 0x56
  lw  $3,48($fp)
  beq $3,$2,$L30
  b $L33
$L35:
  li  $2,111      # 0x6f
  lw  $3,48($fp)
  beq $3,$2,$L32
  li  $2,114      # 0x72
  lw  $3,48($fp)
  beq $3,$2,$L31
  b $L33
$L29:
  la  $25,displayHelp
  jal $31,$25
  li  $2,1      # 0x1
  sw  $2,28($fp)
  b $L23
$L30:
  la  $25,displayVersion
  jal $31,$25
  li  $2,1      # 0x1
  sw  $2,28($fp)
  b $L23
$L31:
  addu  $2,$fp,40
  addu  $3,$fp,44
  lw  $4,optarg
  move  $5,$2
  move  $6,$3
  la  $25,getXY
  jal $31,$25
  b $L23
$L32:
  lw  $2,optarg
  sw  $2,36($fp)
  li  $2,1      # 0x1
  sw  $2,28($fp)
  b $L23
$L33:
  li  $2,1      # 0x1
  sw  $2,28($fp)
  la  $4,$LC4
  la  $25,printf
  jal $31,$25
```

```
  la  $25,displayHelp
  jal $31,$25
  b $L23
$L27:
  li  $2,1      # 0x1
  sw  $2,28($fp)
  la  $4,$LC4
  la  $25,printf
  jal $31,$25
  la  $25,displayHelp
  jal $31,$25
  b $L23
$L24:
  lw  $2,36($fp)
  beq $2,$0,$L37
  lw  $4,40($fp)
  lw  $5,44($fp)
  lw  $6,36($fp)
  la  $25,pgm
  jal $31,$25
$L37:
  move  $2,$0
  move  $sp,$fp
  lw  $31,64($sp)
  lw  $fp,60($sp)
  addu  $sp,$sp,72
  j $31
  .end  main
  .size main, .-main
  .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

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

Código 8: *"pgm.s"*

```
  .file 1 "pgm.c"
  .section .mdebug.abi32
  .previous
  .abicalls
  .rdata
  .align  2
$LC0:
  .ascii  "%d \000"
  .align  2
$LC1:
  .ascii  "\n\000"
  .text
  .align  2
  .globl  escribirLinea
  .ent  escribirLinea
escribirLinea:
  .frame  $fp,64,$31    # vars= 24, regs= 3/0, args= 16, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,64
  .cprestore 16
  sw  $31,56($sp)
  sw  $fp,52($sp)
  sw  $28,48($sp)
  move  $fp,$sp
  sw  $4,64($fp)
  sw  $5,68($fp)
  move  $2,$6
  sb  $2,24($fp)
  lw  $2,64($fp)
  srl $2,$2,3
  sw  $2,32($fp)
```

```
  lbu $2,64($fp)
  andi  $2,$2,0x7
  sb  $2,36($fp)
  sw  $0,40($fp)
  sw  $0,28($fp)
$L6:
  lw  $2,28($fp)
  lw  $3,64($fp)
  sltu  $2,$2,$3
  bne $2,$0,$L9
  b $L7
$L9:
  lb  $2,24($fp)
  lw  $4,68($fp)
  la  $5,$LC0
  move  $6,$2
  la  $25,fprintf
  jal $31,$25
  lw  $2,40($fp)
  addu  $2,$2,1
  sw  $2,40($fp)
  lbu $2,36($fp)
  beq $2,$0,$L11
  lw  $2,32($fp)
  addu  $3,$2,1
  lw  $2,40($fp)
  beq $2,$3,$L12
  b $L8
$L11:
  lw  $3,40($fp)
  lw  $2,32($fp)
  beq $3,$2,$L12
  b $L8
$L12:
  lb  $2,24($fp)
  xori  $2,$2,0x0
  sltu  $2,$2,1
  sb  $2,24($fp)
  lbu $2,36($fp)
  beq $2,$0,$L13
  lbu $2,36($fp)
  addu  $2,$2,-1
  sb  $2,36($fp)
$L13:
  sw  $0,40($fp)
$L8:
  lw  $2,28($fp)
  addu  $2,$2,1
  sw  $2,28($fp)
  b $L6
$L7:
  lw  $4,68($fp)
  la  $5,$LC1
  la  $25,fprintf
  jal $31,$25
  move  $sp,$fp
  lw  $31,56($sp)
  lw  $fp,52($sp)
  addu  $sp,$sp,64
  j $31
  .end  escribirLinea
  .size escribirLinea, .-escribirLinea
  .rdata
  .align  2
$LC2:
  .ascii  "-\000"
  .align  2
$LC3:
  .ascii  "w\000"
  .align  2
$LC4:
  .ascii  "P2\n"
  .ascii  "# %s\n"
  .ascii  "%d %d\n"
  .ascii  "1\n\000"
  .text
```

```
  .align  2
  .globl  pgm
  .ent  pgm
pgm:
  .frame  $fp,72,$31    # vars= 24, regs= 3/0, args= 24, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,72
  .cprestore 24
  sw  $31,64($sp)
  sw  $fp,60($sp)
  sw  $28,56($sp)
  move  $fp,$sp
  sw  $4,72($fp)
  sw  $5,76($fp)
  sw  $6,80($fp)
  li  $2,1      # 0x1
  sb  $2,40($fp)
  lw  $2,76($fp)
  srl $2,$2,3
  sw  $2,44($fp)
  lbu $2,76($fp)
  andi  $2,$2,0x7
  sb  $2,48($fp)
  sw  $0,52($fp)
  lw  $4,80($fp)
  la  $5,$LC2
  la  $25,strcmp
  jal $31,$25
  bne $2,$0,$L15
  la  $2,__sF+88
  sw  $2,32($fp)
  b $L16
$L15:
  lw  $4,80($fp)
  la  $5,$LC3
  la  $25,fopen
  jal $31,$25
  sw  $2,32($fp)
$L16:
  lw  $2,76($fp)
  sw  $2,16($sp)
  lw  $4,32($fp)
  la  $5,$LC4
  lw  $6,80($fp)
  lw  $7,72($fp)
  la  $25,fprintf
  jal $31,$25
  sw  $0,36($fp)
$L17:
  lw  $2,36($fp)
  lw  $3,76($fp)
  sltu  $2,$2,$3
  bne $2,$0,$L20
  b $L18
$L20:
  lb  $2,40($fp)
  lw  $4,72($fp)
  lw  $5,32($fp)
  move  $6,$2
  la  $25,escribirLinea
  jal $31,$25
  lw  $2,52($fp)
  addu  $2,$2,1
  sw  $2,52($fp)
  lbu $2,48($fp)
  beq $2,$0,$L22
  lw  $2,44($fp)
  addu  $3,$2,1
  lw  $2,52($fp)
  beq $2,$3,$L23
  b $L19
$L22:
```

```
  lw  $3,52($fp)
  lw  $2,44($fp)
  beq $3,$2,$L23
  b $L19
$L23:
  lb  $2,40($fp)
  xori  $2,$2,0x0
  sltu  $2,$2,1
  sb  $2,40($fp)
  lbu $2,48($fp)
  beq $2,$0,$L24
  lbu $2,48($fp)
  addu  $2,$2,-1
  sb  $2,48($fp)
$L24:
  sw  $0,52($fp)
$L19:
  lw  $2,36($fp)
  addu  $2,$2,1
  sw  $2,36($fp)
  b $L17
$L18:
  lw  $3,32($fp)
  la  $2,__sF+88
  beq $3,$2,$L14
  lw  $4,32($fp)
  la  $25,fclose
  jal $31,$25
$L14:
  move  $sp,$fp
  lw  $31,64($sp)
  lw  $fp,60($sp)
  addu  $sp,$sp,72
  j $31
  .end  pgm
  .size pgm, .-pgm
  .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```