66.20 Organización de Computadoras
Trabajo Práctico 0:
Infraestructura básica

Burdet Rodrigo, *Padrón Nro. 93440*
rodrigoburdet@gmail.com

Colangelo Federico, *Padrón Nro. 89869*
federico.colangelo@semperti.com

Manzano Matias, *Padrón Nro. 83425*
matsebman@gmail.com

2do. Cuatrimestre de 2014
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

7 de octubre de 2014

1. Objetivos

Familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementando un programa (y su correspondiente documentación) que resuelva el problema piloto que presentaremos mas abajo.

2. Resumen

En el presente trabajo, se implementó un algoritmo que permite graficar los conjuntos de Mandelbrot dados ciertos parámetros para permitir centrarnos en una región en particular de dicho conjunto. El programa fue realizado en c 99 en un entorno de desarrollo linux. Compilado en Linux y en un ambiente emulado NetBSD.

3. Desarrollo

3.1. Paso 1: Configuración de Entorno de Desarrollo

El primer paso fue configurar el entorno de desarrollo, de acuerdo a la guía facilitada por la cátedra. Trabajamos con distribuciones Linux y con el GxEmul proporcionado por la cátedra, emulando un sistema NetBSD.

3.2. Paso 2: Implementación del programa

El programa debe ejecutarse por línea de comando y la salida del mismo dependerá del valor de los argumentos con los que se lo haya invocado.

3.2.1. Ingreso de parámetros

El formato para invocar al programa es el siguiente:

```
./tp0 [OPTIONS]
```

Los parámetros válidos que puede recibir el programa son los siguientes:

-o,	**–output**	`(Parámetro obligatorio. Especifica archivo de salida, - para stdou`
-r,	**–resolution**	`(Resolución de la imagen de salida).`
-c,	**–center**	`(Centro de la imagen).`
-w,	**–width**	`(Ancho del rectángulo a dibujar).`
-H,	**–height**	`(Alto del rectángulo a dibujar).`
-v,	**–version**	`(Muestra la versión).`
-h,	**–help**	`(Muestra la ayuda).`

3.2.2. Interpretación de parámetros

Para parsear los parámetros se usó la librería de GNU getopt, en particular se usó `getopt_long` para permitir el pasaje de parámetros largos.

4. Compilación del programa

Para poder compilar el proyecto, se debe abrir una terminal Linux dentro del directorio donde se encuentra el código fuente escrito en C, y ejecutar el siguiente comando:

```
gcc -Wall -std=c99 main.c -o tp0
```
[1]

Esto generara un archivo ejecutable, llamado *tp0*.

[1] Requiere tener instalado el compilador *GCC*

5. Compilación del programa en NetBSD

Para poder compilar el proyecto en NetBSD, se debe ejecutar el comando:

```
gcc -Wall -std=c99 -S -O0 main.c
```

6. Corridas de prueba y Mediciones

En las figuras que siguen a continuación se muestran los comandos utilizados para ejecutar el programa y se puede apreciar los resultados de las diferentes pruebas que realizamos.

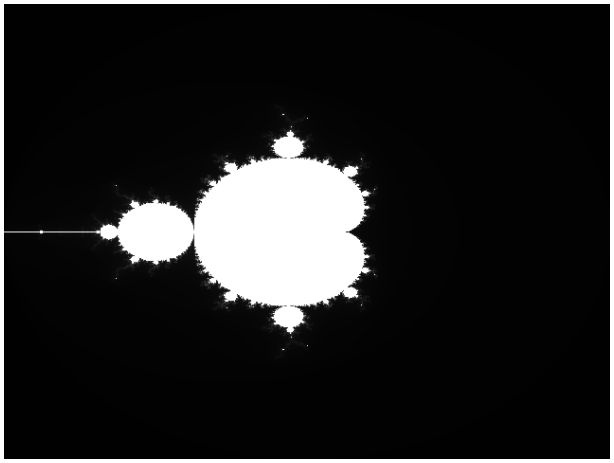

Figura 1: Llamada por defecto ./tp0 -o uno.pgm

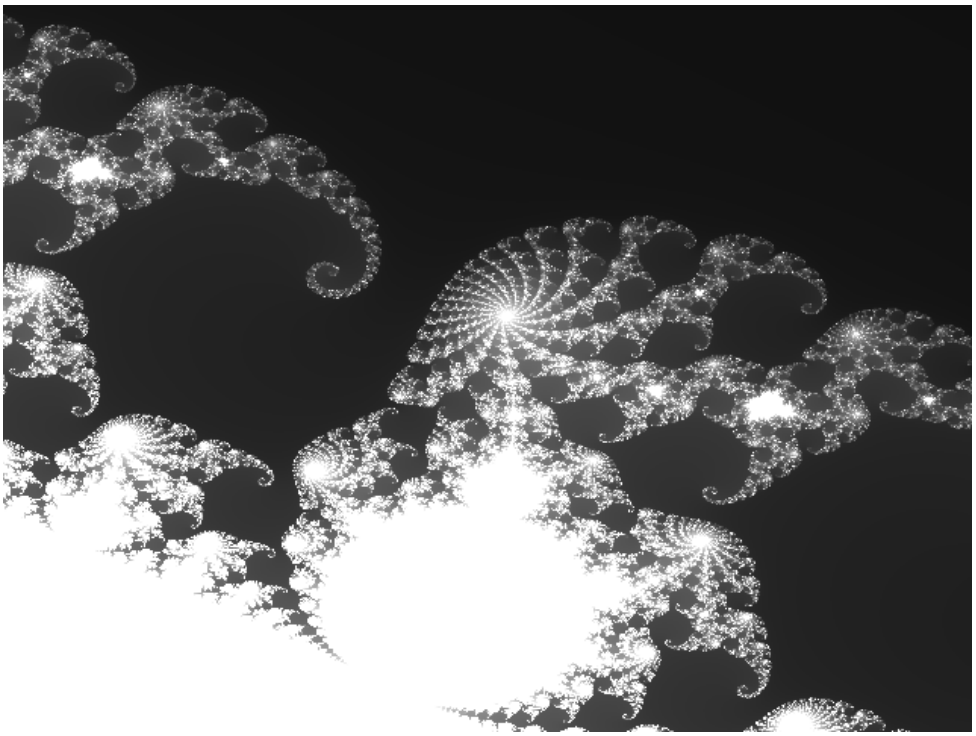

Figura 2: Llamada haciendo zoom sobre la región centrada en (0.282 , -0.01) con cuadrado de 0.005 de lado. ./tp0 -c +0.282-0.01i -w 0.005 -H 0.005 -o dos.pgm

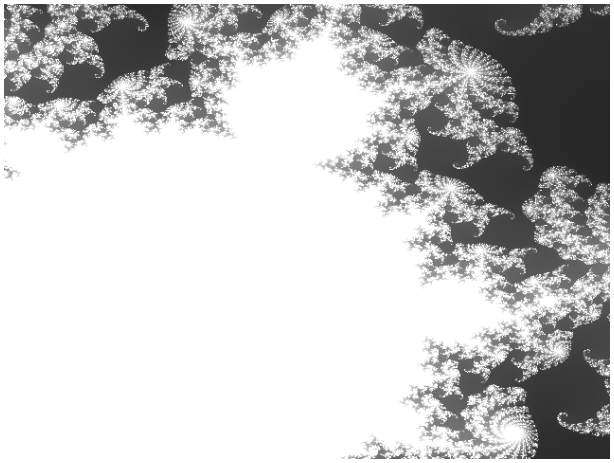

Figura 3: Llamada haciendo zoom sobre la región centrada en (0.296 , -0.02) con cuadrado de 0.003 de lado. ./tp0 -c +0.296-0.02i -w 0.003 -H 0.003 -o tres.pgm

7. Conclusiones

Como se enuncia en el objetivo de este trabajo práctico, aprendimos a instalar y manejar el GxEmul, a realizar transferencias de archivos en Linux, así como también compilar y ejecutar programas en el NetBSD. Por otro lado, aprendimos a manejar y escribir informes en LaTeX. De este modo, estamos preparados para que en los próximos trabajos prácticos, nos aboquemos directamente al desarrollo de los mismos.

8. Código

8.1. main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <string.h>

#define MAX_VAL 255
#define USAGE_ERROR -1
#define IO_ERROR -2
static const char VERSION[] = "1.0.1";

void usage(char* nombre);
void version(char* nombre);

/** Dada una resolucion en formato WIDTHxHEIGHT devuelve un vector [WIDTH,HEIGHT] por defecto es [640,480]
 * @param char* String con resolucion del tipo WIDTHxHEIGHT
 * @param int* vector de ints de resolucion [WIDTH,HEIGHT]
 * @return int 0 si se pudo parsear la resolucion, -1 en otro caso
 */

int parseResolution(char* res,int* resArray);

/** Dada una coordenada compleja en representacion binomica ( a + bi ) devuelve un vector [posX, posY], por defecto es [0,0]
 * @param char* String con posicion del centro de pantalla en forma binomica a+bi
 * @param double* vector de ints de centro [posX,posY]
 * @return int 0 si se pudo parsear la resolucion, -1 en otro caso
 */

int parseCenter(char* centerString, double* center);

/** Dadas las partes real e imaginara de un numero complejo devuelve 1 si el modulo es mayor que 2, 0 en caso contrario
 * @param double parte real de un numero complejo
 * @param double parte imaginaria de un numero complejo
 * return int 1 si es el modulo es mayor que 2, 0 si es menor o igual
 */

int stop(double re, double im);

/** Dados los parametros de interes calcula los conjuntos de mandelbrot
 * @param int* vector de 2 enteros para especificar la resolucion
 * @param double* vector de 2 doubles para especificar la coordenada de origen
 * @param double ancho del rectangulo a graficar
 * @param double alto del rectangulo a graficar
 * @param FILE* archivo donde escribe los conjuntos en formato pgm
 * return int 0 si no hubo error
 */
int print(int* res, double* center, double width , double height,FILE* file);


int main(int argc, char* argv[]){
  int opt;
  int res[2] = {640,480};
  double  width = 4;
  double  height = 4;
  double center[2] = {0,0};
  char* outDir = NULL;
  FILE* file = stdout;
  int parseCenterResult = 0;
  int parseResolutionResult = 0;
```

```
int noArguments = 0;

static struct option long_options[] ={
  {"version",    no_argument,       0,  'V'},
  {"help",     no_argument,       0,  'h'},
  {"resolution",  required_argument,     0,  'r'},
  {"center",     required_argument,     0,  'c'},
  {"width",    required_argument,     0,  'w'},
  {"height",     required_argument,     0,  'H'},
  {"output",     required_argument,     0,  'o'},
  {0, 0, 0, 0}
};

while ((opt = getopt_long (argc, argv, "r:c:w:H:o:Vh", long_options, NULL)) != -1){
  switch(opt){
    case 'r':
      noArguments++;
      parseResolutionResult = parseResolution(optarg, res);
      break;
    case 'w':
      noArguments++;
      width = atof(optarg);
      break;
    case 'H':
      noArguments++;
      height = atof(optarg);
      break;
    case 'o':
      noArguments++;
      outDir = optarg;
      break;
    case 'c':
      noArguments++;
      parseCenterResult = parseCenter(optarg, center);
      break;
    case 'h':
      usage(argv[0]);
      return 0;
    case 'V':
      version(argv[0]);
      return 0;
    case '?':
      printf("Error\n");
      return USAGE_ERROR;
  }
}

if (noArguments == 0) {
  usage(argv[0]);
  return 0;
}

if (parseCenterResult == -1){
  usage(argv[0]);
  if (fprintf(stderr, "fatal: invalid center specification.\n") < 0)
    return IO_ERROR;
  return USAGE_ERROR;
}

if (parseResolutionResult == -1){
  usage(argv[0]);
  if (fprintf(stderr, "fatal: invalid resolution specification.\n") < 0)
    return IO_ERROR;
  return USAGE_ERROR;
}

if (outDir == NULL){
  return USAGE_ERROR;
}
```

```
128
129   if (strcmp(outDir,"-") != 0){
130     file = fopen(outDir,"w");
131     if (!file){
132       usage(argv[0]);
133       if (fprintf(stderr, "fatal: cannot open output file.\n") < 1)
134         return IO_ERROR;
135       return USAGE_ERROR;
136     }
137   }
138
139   if (res[0] <= 0 || res[1] <= 0)
140     usage(argv[0]);
141   else{
142     if (fprintf(file, "P2\n %d\n %d\n %d\n", res[0],res[1],MAX_VAL) < 0)
143       return IO_ERROR;
144       print(res,center,width,height,file);
145   }
146
147   if (file != stdout)
148     if (fclose(file) < 0)
149       return IO_ERROR;
150
151   return 0;
152 }
153
154 int print(int* res, double* center, double width , double height,FILE* file){
155   double stepX = width / res[0];
156   double stepY = height/ res[1];
157   int i, j, k;
158   double x, y;
159   double zx, zy;
160   double zx2, zy2;
161   x= center[0] + (-width+stepX)/2;
162   //y= center[1] + (-height+stepY)/2;
163   y= center[1] + (height-stepY)/2;
164   for ( i = 1 ; i <= res[1] ; ++i ) {
165     //y = center[1]+height/2 - i*stepY;
166       //printf("punto : %f , %f \n",x,y);
167     for ( j = 1 ; j <= res[0] ; ++j) {
168       //x = center[0]-width/2 + j*stepX;
169
170
171       zx = 0;
172       zy = 0;
173       zx2 = zx*zx;
174       zy2 = zy*zy;
175       for ( k = -1 ; k < MAX_VAL && !stop(zx,zy); k++ ){
176         zy = 2*zx*zy + y;
177         zx = zx2 - zy2 + x;
178         zx2 = zx*zx;
179         zy2 = zy*zy;
180         //printf("calculo : %f , %f \n",zx,zy);
181       }
182       if (fprintf(file,"%d",k) < 0)
183         return IO_ERROR;
184       if (fputc(' ',file) < 0)
185         return IO_ERROR;
186       x= center[0] + (-width+2*j*stepX)/2;
187     }
188     y= center[1] +(height-2*i*stepY)/2;
189     if (fputc('\n',file) < 0)
190       return IO_ERROR;
191   }
192   return 0;
193 }
194
195 int parseResolution(char* str, int* res){
196   if (strlen(str) < 3)
```

```
    return USAGE_ERROR;
  char* aux ;
  if ( ( aux = strtok(str,"x")) != NULL){
    res[0] = atoi(aux);
    aux = strtok(NULL,"x");
    res[1] = atoi(aux);
  }else
    return USAGE_ERROR;
  return 0;
}

int parseCenter(char* centerString, double* center){
  /*
  if (str[strlen(str)-1] != 'i'){
    return USAGE_ERROR;
  }
  char* aux = calloc(strlen(str)+1,sizeof(char));
  unsigned int i = 0;
  int pos;
  int real = 1;
  int signR;
  int signI = 1;
  if(str[0] == '+'|| str[0] != '-'){
    signR = 1 ;
    i++;
  }
  if(str[0]=='-')
    signR = -1;
  for ( ; i< strlen(str) ; i++){
    if (real){
      aux[i-1] = str[i];
      if (str[i] == '-' || str[i] == '+'){
        pos = i;
        center[0]= signR * atof(aux);
        real = 0;
        memset(aux,'\0',strlen(str));
      }
    }else{
      if (str[pos] == '-')
        signI = -1;
      aux[i-pos-1] = str[i];
    }
  }
  aux[strlen(aux)-1] = '\0';
  center[1] = signI *atof(aux);
  return 0;

  //////////////////////////

  if (centerString[strlen(centerString)-1] != 'i'){
    return USAGE_ERROR;
  }
  center[0] = atof(strtok(centerString, "+i"));
  center[1] = atof(strtok(NULL, "+i"));
  return 0;
  */
  int realSign = 1;
  int complexSign = 1;
  unsigned int i = 1;
  unsigned int found = 0;
  if (centerString[strlen(centerString)-1] != 'i') {
    return USAGE_ERROR;
  }
  if (centerString[0] == '-') {
    realSign = -1;
  }
  while (found == 0 && i < strlen(centerString)) {
    if (centerString[i] == '+') {
      found = 1;
```

```
266       }
267       else if (centerString[i] == '-') {
268         found = 1;
269         complexSign = -1;
270       }
271       else {
272         i++;
273       }
274     }
275     if (found == 0) {
276       return USAGE_ERROR;
277     }
278     center[0] = atof(strtok(centerString, "+-i")) * realSign;
279     center[1] = atof(strtok(NULL, "+-i")) * complexSign;
280     return 0;
281   }
282
283
284   void version(char* name){
285     printf("%s version: %s: \n",name, VERSION);
286   }
287
288   int stop(double re, double im){
289     if ((re*re)+(im*im)>4)
290       return 1;
291     return 0;
292   }
293
294   void usage(char* name){
295     printf("Usage: \n");
296     printf("\t %s -h --help \n",name);
297     printf("\t %s -V --version \n",name);
298     printf("\t %s -r --resolution, permite cambiar la resolucion de la imagen generada. El ↩
          valor por defecto sera de 640x480 puntos \n",name);
299     printf("\t %s -c --center, para especificar el centro de la imagen, el punto central de ↩
          la porcion del plano complejo dibujada, expresado en forma binomica. El valor por ↩
          defecto sera 0 + 0i  \n",name);
300     printf("\t %s -w --width, especifica el ancho del rectangulo que contiene la region del ↩
          plano complejo que estamos por dibujar. El valor por defecto sera 4 \n",name);
301     printf("\t %s -H --height, sirve, en forma similar, para especificar el alto del ↩
          rectangulo a dibujar. El valor por defecto sera 4 \n",name);
302     printf("\t %s -o --output, permite colocar la imagen de salida, (en formato PGM) en el ↩
          archivo pasado como argumento; o por salida estandar -cout- si el argumento es \"-\" ↩
          \n",name);
303     printf("Examples: \n");
304     printf("\t %s -o uno.pgm \n",name);
305     printf("\t %s -c +0.282-0.01i -w 0.005 -H 0.005 -o dos.pgm \n",name);
306     printf("\t %s -r 1x1 -o - \n",name);
307   }
```

8.2. main.s

```
  .file 1 "main.c"
  .section .mdebug.abi32
  .previous
  .abicalls
  .rdata
  .align  2
  .type VERSION, @object
  .size VERSION, 6
VERSION:
  .ascii  "1.0.1\000"
  .align  2
$LC0:
  .word 640
  .word 480
  .align  2
$LC2:
  .ascii  "version\000"
  .align  2
$LC3:
  .ascii  "help\000"
  .align  2
$LC4:
  .ascii  "resolution\000"
  .align  2
$LC5:
  .ascii  "center\000"
  .align  2
$LC6:
  .ascii  "width\000"
  .align  2
$LC7:
  .ascii  "height\000"
  .align  2
$LC8:
  .ascii  "output\000"
  .data
  .align  2
  .type long_options.0, @object
  .size long_options.0, 128
long_options.0:
  .word $LC2
  .word 0
  .word 0
  .word 86
  .word $LC3
  .word 0
  .word 0
  .word 104
  .word $LC4
  .word 1
  .word 0
  .word 114
  .word $LC5
  .word 1
  .word 0
  .word 99
  .word $LC6
  .word 1
  .word 0
  .word 119
  .word $LC7
  .word 1
  .word 0
  .word 72
```

```
  .word $LC8
  .word 1
  .word 0
  .word 111
  .word 0
  .word 0
  .word 0
  .word 0
  .rdata
  .align  2
$LC9:
  .ascii  "r:c:w:H:o:Vh\000"
  .align  2
$LC10:
  .ascii  "Error\n\000"
  .align  2
$LC11:
  .ascii  "fatal: invalid center specification.\n\000"
  .align  2
$LC12:
  .ascii  "fatal: invalid resolution specification.\n\000"
  .align  2
$LC13:
  .ascii  "-\000"
  .align  2
$LC14:
  .ascii  "w\000"
  .align  2
$LC15:
  .ascii  "fatal: cannot open output file.\n\000"
  .align  2
$LC16:
  .ascii  "P2\n"
  .ascii  "%d\n"
  .ascii  "%d\n"
  .ascii  "%d\n\000"
  .align  3
$LC1:
  .word 0
  .word 1074790400
  .text
  .align  2
  .globl  main
  .ent  main
main:
  .frame  $fp,136,$31   # vars= 80, regs= 3/0, args= 32, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,136
  .cprestore 32
  sw  $31,128($sp)
  sw  $fp,124($sp)
  sw  $28,120($sp)
  move  $fp,$sp
  sw  $4,136($fp)
  sw  $5,140($fp)
  lw  $2,$LC0
  sw  $2,48($fp)
  lw  $2,$LC0+4
  sw  $2,52($fp)
  l.d $f0,$LC1
  s.d $f0,56($fp)
  l.d $f0,$LC1
  s.d $f0,64($fp)
  sw  $0,72($fp)
  sw  $0,76($fp)
```

```
  sw  $0,80($fp)
  sw  $0,84($fp)
  sw  $0,88($fp)
  la  $2,__sF+88
  sw  $2,92($fp)
  sw  $0,96($fp)
  sw  $0,100($fp)
  sw  $0,104($fp)
$L18:
  sw  $0,16($sp)
  lw  $4,136($fp)
  lw  $5,140($fp)
  la  $6,$LC9
  la  $7,long_options.0
  la  $25,getopt_long
  jal $31,$25
  sw  $2,40($fp)
  lw  $3,40($fp)
  li  $2,-1     # 0xffffffffffffffff
  bne $3,$2,$L20
  b $L19
$L20:
  lw  $2,40($fp)
  addu  $2,$2,-63
  sw  $2,112($fp)
  lw  $3,112($fp)
  sltu  $2,$3,57
  beq $2,$0,$L18
  lw  $2,112($fp)
  sll $3,$2,2
  la  $2,$L30
  addu  $2,$3,$2
  lw  $2,0($2)
  .cpadd  $2
  j $2
  .rdata
  .align  2
$L30:
  .gpword $L29
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L24
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L28
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
```

```
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L26
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L27
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L25
  .gpword $L18
  .gpword $L18
  .gpword $L22
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L18
  .gpword $L23
  .text
$L22:
  lw  $2,104($fp)
  addu  $2,$2,1
  sw  $2,104($fp)
  addu  $2,$fp,48
  lw  $4,optarg
  move  $5,$2
  la  $25,parseResolution
  jal $31,$25
  sw  $2,100($fp)
  b $L18
$L23:
  lw  $2,104($fp)
  addu  $2,$2,1
  sw  $2,104($fp)
  lw  $4,optarg
  la  $25,atof
  jal $31,$25
  s.d $f0,56($fp)
  b $L18
$L24:
  lw  $2,104($fp)
  addu  $2,$2,1
  sw  $2,104($fp)
  lw  $4,optarg
  la  $25,atof
  jal $31,$25
  s.d $f0,64($fp)
  b $L18
$L25:
  lw  $2,104($fp)
  addu  $2,$2,1
  sw  $2,104($fp)
  lw  $2,optarg
  sw  $2,88($fp)
  b $L18
$L26:
  lw  $2,104($fp)
  addu  $2,$2,1
  sw  $2,104($fp)
  addu  $2,$fp,72
  lw  $4,optarg
```

```
  move  $5,$2
  la  $25,parseCenter
  jal $31,$25
  sw  $2,96($fp)
  b $L18
$L27:
  lw  $2,140($fp)
  lw  $4,0($2)
  la  $25,usage
  jal $31,$25
  sw  $0,108($fp)
  b $L17
$L28:
  lw  $2,140($fp)
  lw  $4,0($2)
  la  $25,version
  jal $31,$25
  sw  $0,108($fp)
  b $L17
$L29:
  la  $4,$LC10
  la  $25,printf
  jal $31,$25
  li  $2,-1     # 0xffffffffffffffff
  sw  $2,108($fp)
  b $L17
$L19:
  lw  $2,104($fp)
  bne $2,$0,$L32
  lw  $2,140($fp)
  lw  $4,0($2)
  la  $25,usage
  jal $31,$25
  sw  $0,108($fp)
  b $L17
$L32:
  lw  $3,96($fp)
  li  $2,-1     # 0xffffffffffffffff
  bne $3,$2,$L33
  lw  $2,140($fp)
  lw  $4,0($2)
  la  $25,usage
  jal $31,$25
  la  $4,__sF+176
  la  $5,$LC11
  la  $25,fprintf
  jal $31,$25
  bgez  $2,$L34
  li  $3,-2     # 0xfffffffffffffffe
  sw  $3,108($fp)
  b $L17
$L34:
  li  $2,-1     # 0xffffffffffffffff
  sw  $2,108($fp)
  b $L17
$L33:
  lw  $3,100($fp)
  li  $2,-1     # 0xffffffffffffffff
  bne $3,$2,$L35
  lw  $2,140($fp)
  lw  $4,0($2)
  la  $25,usage
  jal $31,$25
  la  $4,__sF+176
  la  $5,$LC12
  la  $25,fprintf
  jal $31,$25
  bgez  $2,$L36
  li  $3,-2     # 0xfffffffffffffffe
```

```
  sw  $3,108($fp)
  b $L17
$L36:
  li  $2,-1     # 0xffffffffffffffff
  sw  $2,108($fp)
  b $L17
$L35:
  lw  $2,88($fp)
  bne $2,$0,$L37
  li  $3,-1     # 0xffffffffffffffff
  sw  $3,108($fp)
  b $L17
$L37:
  lw  $4,88($fp)
  la  $5,$LC13
  la  $25,strcmp
  jal $31,$25
  beq $2,$0,$L38
  lw  $4,88($fp)
  la  $5,$LC14
  la  $25,fopen
  jal $31,$25
  sw  $2,92($fp)
  lw  $2,92($fp)
  bne $2,$0,$L38
  lw  $2,140($fp)
  lw  $4,0($2)
  la  $25,usage
  jal $31,$25
  la  $4,__sF+176
  la  $5,$LC15
  la  $25,fprintf
  jal $31,$25
  bgtz  $2,$L40
  li  $2,-2     # 0xfffffffffffffffe
  sw  $2,108($fp)
  b $L17
$L40:
  li  $3,-1     # 0xffffffffffffffff
  sw  $3,108($fp)
  b $L17
$L38:
  lw  $2,48($fp)
  blez  $2,$L42
  lw  $2,52($fp)
  blez  $2,$L42
  b $L41
$L42:
  lw  $2,140($fp)
  lw  $4,0($2)
  la  $25,usage
  jal $31,$25
  b $L43
$L41:
  li  $2,255      # 0xff
  sw  $2,16($sp)
  lw  $4,92($fp)
  la  $5,$LC16
  lw  $6,48($fp)
  lw  $7,52($fp)
  la  $25,fprintf
  jal $31,$25
  bgez  $2,$L44
  li  $2,-2     # 0xfffffffffffffffe
  sw  $2,108($fp)
  b $L17
$L44:
  addu  $3,$fp,48
  addu  $5,$fp,72
```

```
  l.d $f0,64($fp)
  s.d $f0,16($sp)
  lw  $2,92($fp)
  sw  $2,24($sp)
  move  $4,$3
  lw  $6,56($fp)
  lw  $7,60($fp)
  la  $25,print
  jal $31,$25
$L43:
  lw  $3,92($fp)
  la  $2,__sF+88
  beq $3,$2,$L45
  lw  $4,92($fp)
  la  $25,fclose
  jal $31,$25
  bgez  $2,$L45
  li  $3,-2     # 0xfffffffffffffffe
  sw  $3,108($fp)
  b $L17
$L45:
  sw  $0,108($fp)
$L17:
  lw  $2,108($fp)
  move  $sp,$fp
  lw  $31,128($sp)
  lw  $fp,124($sp)
  addu  $sp,$sp,136
  j $31
  .end  main
  .size main, .-main
  .rdata
  .align  2
$LC18:
  .ascii  "%d\000"
  .align  3
$LC17:
  .word 0
  .word 1073741824
  .text
  .align  2
  .globl  print
  .ent  print
print:
  .frame  $fp,128,$31   # vars= 88, regs= 3/0, args= 16, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,128
  .cprestore 16
  sw  $31,120($sp)
  sw  $fp,116($sp)
  sw  $28,112($sp)
  move  $fp,$sp
  sw  $4,128($fp)
  sw  $5,132($fp)
  sw  $6,136($fp)
  sw  $7,140($fp)
  lw  $2,128($fp)
  l.s $f0,0($2)
  cvt.d.w $f2,$f0
  l.d $f0,136($fp)
  div.d $f0,$f0,$f2
  s.d $f0,24($fp)
  lw  $2,128($fp)
  addu  $2,$2,4
  l.s $f0,0($2)
```

```
  cvt.d.w $f2,$f0
  l.d $f0,144($fp)
  div.d $f0,$f0,$f2
  s.d $f0,32($fp)
  lw  $2,132($fp)
  l.d $f2,24($fp)
  l.d $f0,136($fp)
  sub.d $f2,$f2,$f0
  l.d $f0,$LC17
  div.d $f2,$f2,$f0
  l.d $f0,0($2)
  add.d $f0,$f0,$f2
  s.d $f0,56($fp)
  lw  $2,132($fp)
  addu  $2,$2,8
  l.d $f2,144($fp)
  l.d $f0,32($fp)
  sub.d $f2,$f2,$f0
  l.d $f0,$LC17
  div.d $f2,$f2,$f0
  l.d $f0,0($2)
  add.d $f0,$f0,$f2
  s.d $f0,64($fp)
  li  $2,1      # 0x1
  sw  $2,40($fp)
$L48:
  lw  $2,128($fp)
  addu  $2,$2,4
  lw  $3,40($fp)
  lw  $2,0($2)
  slt $2,$2,$3
  beq $2,$0,$L51
  b $L49
$L51:
  li  $2,1      # 0x1
  sw  $2,44($fp)
$L52:
  lw  $2,128($fp)
  lw  $3,44($fp)
  lw  $2,0($2)
  slt $2,$2,$3
  beq $2,$0,$L55
  b $L53
$L55:
  sw  $0,72($fp)
  sw  $0,76($fp)
  sw  $0,80($fp)
  sw  $0,84($fp)
  l.d $f2,72($fp)
  l.d $f0,72($fp)
  mul.d $f0,$f2,$f0
  s.d $f0,88($fp)
  l.d $f2,80($fp)
  l.d $f0,80($fp)
  mul.d $f0,$f2,$f0
  s.d $f0,96($fp)
  li  $2,-1     # 0xffffffffffffffff
  sw  $2,48($fp)
$L56:
  lw  $2,48($fp)
  slt $2,$2,255
  beq $2,$0,$L57
  l.d $f12,72($fp)
  l.d $f14,80($fp)
  la  $25,stop
  jal $31,$25
  bne $2,$0,$L57
  l.d $f0,72($fp)
  add.d $f2,$f0,$f0
```

```
  l.d $f0,80($fp)
  mul.d $f2,$f2,$f0
  l.d $f0,64($fp)
  add.d $f0,$f2,$f0
  s.d $f0,80($fp)
  l.d $f2,88($fp)
  l.d $f0,96($fp)
  sub.d $f2,$f2,$f0
  l.d $f0,56($fp)
  add.d $f0,$f2,$f0
  s.d $f0,72($fp)
  l.d $f2,72($fp)
  l.d $f0,72($fp)
  mul.d $f0,$f2,$f0
  s.d $f0,88($fp)
  l.d $f2,80($fp)
  l.d $f0,80($fp)
  mul.d $f0,$f2,$f0
  s.d $f0,96($fp)
  lw  $2,48($fp)
  addu  $2,$2,1
  sw  $2,48($fp)
  b $L56
$L57:
  lw  $4,152($fp)
  la  $5,$LC18
  lw  $6,48($fp)
  la  $25,fprintf
  jal $31,$25
  bgez  $2,$L61
  li  $2,-2     # 0xfffffffffffffffe
  sw  $2,104($fp)
  b $L47
$L61:
  li  $4,32     # 0x20
  lw  $5,152($fp)
  la  $25,fputc
  jal $31,$25
  bgez  $2,$L62
  li  $2,-2     # 0xfffffffffffffffe
  sw  $2,104($fp)
  b $L47
$L62:
  lw  $3,132($fp)
  lw  $2,44($fp)
  sll $2,$2,1
  mtc1  $2,$f0
  cvt.d.w $f2,$f0
  l.d $f0,24($fp)
  mul.d $f2,$f2,$f0
  l.d $f0,136($fp)
  sub.d $f2,$f2,$f0
  l.d $f0,$LC17
  div.d $f2,$f2,$f0
  l.d $f0,0($3)
  add.d $f0,$f0,$f2
  s.d $f0,56($fp)
  lw  $2,44($fp)
  addu  $2,$2,1
  sw  $2,44($fp)
  b $L52
$L53:
  lw  $2,132($fp)
  addu  $3,$2,8
  lw  $2,40($fp)
  sll $2,$2,1
  mtc1  $2,$f0
  cvt.d.w $f2,$f0
  l.d $f0,32($fp)
```

```
  mul.d $f2,$f2,$f0
  l.d $f0,144($fp)
  sub.d $f2,$f0,$f2
  l.d $f0,$LC17
  div.d $f2,$f2,$f0
  l.d $f0,0($3)
  add.d $f0,$f0,$f2
  s.d $f0,64($fp)
  li  $4,10     # 0xa
  lw  $5,152($fp)
  la  $25,fputc
  jal $31,$25
  bgez  $2,$L50
  li  $2,-2     # 0xfffffffffffffffe
  sw  $2,104($fp)
  b $L47
$L50:
  lw  $2,40($fp)
  addu  $2,$2,1
  sw  $2,40($fp)
  b $L48
$L49:
  sw  $0,104($fp)
$L47:
  lw  $2,104($fp)
  move  $sp,$fp
  lw  $31,120($sp)
  lw  $fp,116($sp)
  addu  $sp,$sp,128
  j $31
  .end  print
  .size print, .-print
  .rdata
  .align  2
$LC19:
  .ascii  "x\000"
  .text
  .align  2
  .globl  parseResolution
  .ent  parseResolution
parseResolution:
  .frame  $fp,48,$31    # vars= 8, regs= 3/0, args= 16, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,48
  .cprestore 16
  sw  $31,40($sp)
  sw  $fp,36($sp)
  sw  $28,32($sp)
  move  $fp,$sp
  sw  $4,48($fp)
  sw  $5,52($fp)
  lw  $4,48($fp)
  la  $25,strlen
  jal $31,$25
  sltu  $2,$2,3
  beq $2,$0,$L65
  li  $2,-1     # 0xffffffffffffffff
  sw  $2,28($fp)
  b $L64
$L65:
  lw  $4,48($fp)
  la  $5,$LC19
  la  $25,strtok
  jal $31,$25
  sw  $2,24($fp)
```

```
  lw  $2,24($fp)
  beq $2,$0,$L66
  lw  $4,24($fp)
  la  $25,atoi
  jal $31,$25
  move  $3,$2
  lw  $2,52($fp)
  sw  $3,0($2)
  move  $4,$0
  la  $5,$LC19
  la  $25,strtok
  jal $31,$25
  sw  $2,24($fp)
  lw  $4,24($fp)
  la  $25,atoi
  jal $31,$25
  move  $3,$2
  lw  $2,52($fp)
  addu  $2,$2,4
  sw  $3,0($2)
  b $L67
$L66:
  li  $2,-1     # 0xffffffffffffffff
  sw  $2,28($fp)
  b $L64
$L67:
  sw  $0,28($fp)
$L64:
  lw  $2,28($fp)
  move  $sp,$fp
  lw  $31,40($sp)
  lw  $fp,36($sp)
  addu  $sp,$sp,48
  j $31
  .end  parseResolution
  .size parseResolution, .-parseResolution
  .rdata
  .align  2
$LC20:
  .ascii  "+-i\000"
  .text
  .align  2
  .globl  parseCenter
  .ent  parseCenter
parseCenter:
  .frame  $fp,64,$31    # vars= 24, regs= 4/0, args= 16, extra= 8
  .mask 0xd0010000,-4
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,64
  .cprestore 16
  sw  $31,60($sp)
  sw  $fp,56($sp)
  sw  $28,52($sp)
  sw  $16,48($sp)
  move  $fp,$sp
  sw  $4,64($fp)
  sw  $5,68($fp)
  li  $2,1      # 0x1
  sw  $2,24($fp)
  li  $2,1      # 0x1
  sw  $2,28($fp)
  li  $2,1      # 0x1
  sw  $2,32($fp)
  sw  $0,36($fp)
  lw  $4,64($fp)
  la  $25,strlen
```

```
  jal $31,$25
  move  $3,$2
  lw  $2,64($fp)
  addu  $2,$3,$2
  addu  $2,$2,-1
  lb  $3,0($2)
  li  $2,105      # 0x69
  beq $3,$2,$L69
  li  $2,-1     # 0xffffffffffffffff
  sw  $2,40($fp)
  b $L68
$L69:
  lw  $2,64($fp)
  lb  $3,0($2)
  li  $2,45     # 0x2d
  bne $3,$2,$L70
  li  $2,-1     # 0xffffffffffffffff
  sw  $2,24($fp)
$L70:
  .set  noreorder
  nop
  .set  reorder
$L71:
  lw  $2,36($fp)
  bne $2,$0,$L72
  lw  $4,64($fp)
  la  $25,strlen
  jal $31,$25
  lw  $3,32($fp)
  sltu  $2,$3,$2
  bne $2,$0,$L73
  b $L72
$L73:
  lw  $3,64($fp)
  lw  $2,32($fp)
  addu  $2,$3,$2
  lb  $3,0($2)
  li  $2,43     # 0x2b
  bne $3,$2,$L75
  li  $2,1      # 0x1
  sw  $2,36($fp)
  b $L71
$L75:
  lw  $3,64($fp)
  lw  $2,32($fp)
  addu  $2,$3,$2
  lb  $3,0($2)
  li  $2,45     # 0x2d
  bne $3,$2,$L77
  li  $2,1      # 0x1
  sw  $2,36($fp)
  li  $2,-1     # 0xffffffffffffffff
  sw  $2,28($fp)
  b $L71
$L77:
  lw  $2,32($fp)
  addu  $2,$2,1
  sw  $2,32($fp)
  b $L71
$L72:
  lw  $2,36($fp)
  bne $2,$0,$L79
  li  $2,-1     # 0xffffffffffffffff
  sw  $2,40($fp)
  b $L68
$L79:
  lw  $16,68($fp)
  lw  $4,64($fp)
  la  $5,$LC20
```

```
  la  $25,strtok
  jal $31,$25
  move  $4,$2
  la  $25,atof
  jal $31,$25
  mov.d $f2,$f0
  l.s $f0,24($fp)
  cvt.d.w $f0,$f0
  mul.d $f0,$f2,$f0
  s.d $f0,0($16)
  lw  $2,68($fp)
  addu  $16,$2,8
  move  $4,$0
  la  $5,$LC20
  la  $25,strtok
  jal $31,$25
  move  $4,$2
  la  $25,atof
  jal $31,$25
  mov.d $f2,$f0
  l.s $f0,28($fp)
  cvt.d.w $f0,$f0
  mul.d $f0,$f2,$f0
  s.d $f0,0($16)
  sw  $0,40($fp)
$L68:
  lw  $2,40($fp)
  move  $sp,$fp
  lw  $31,60($sp)
  lw  $fp,56($sp)
  lw  $16,48($sp)
  addu  $sp,$sp,64
  j $31
  .end  parseCenter
  .size parseCenter, .-parseCenter
  .rdata
  .align  2
$LC21:
  .ascii  "%s version: %s: \n\000"
  .text
  .align  2
  .globl  version
  .ent  version
version:
  .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
  sw  $4,40($fp)
  la  $4,$LC21
  lw  $5,40($fp)
  la  $6,VERSION
  la  $25,printf
  jal $31,$25
  move  $sp,$fp
  lw  $31,32($sp)
  lw  $fp,28($sp)
  addu  $sp,$sp,40
  j $31
  .end  version
  .size version, .-version
```

```
  .rdata
  .align  3
$LC22:
  .word 0
  .word 1074790400
  .text
  .align  2
  .globl  stop
  .ent  stop
stop:
  .frame  $fp,24,$31    # vars= 8, regs= 2/0, args= 0, extra= 8
  .mask 0x50000000,-4
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,24
  .cprestore 0
  sw  $fp,20($sp)
  sw  $28,16($sp)
  move  $fp,$sp
  s.d $f12,24($fp)
  s.d $f14,32($fp)
  l.d $f2,24($fp)
  l.d $f0,24($fp)
  mul.d $f4,$f2,$f0
  l.d $f2,32($fp)
  l.d $f0,32($fp)
  mul.d $f0,$f2,$f0
  add.d $f2,$f4,$f0
  l.d $f0,$LC22
  c.lt.d  $f0,$f2
  bc1t  $L83
  b $L82
$L83:
  li  $2,1      # 0x1
  sw  $2,8($fp)
  b $L81
$L82:
  sw  $0,8($fp)
$L81:
  lw  $2,8($fp)
  move  $sp,$fp
  lw  $fp,20($sp)
  addu  $sp,$sp,24
  j $31
  .end  stop
  .size stop, .-stop
  .rdata
  .align  2
$LC23:
  .ascii  "Usage: \n\000"
  .align  2
$LC24:
  .ascii  "\t %s -h --help \n\000"
  .align  2
$LC25:
  .ascii  "\t %s -V --version \n\000"
  .align  2
$LC26:
  .ascii  "\t %s -r --resolution, permite cambiar la resolucion de "
  .ascii  "la imagen generada. El valor por defecto sera de 640x480"
  .ascii  " puntos \n\000"
  .align  2
$LC27:
  .ascii  "\t %s -c --center, para especificar el centro de la imag"
  .ascii  "en, el punto central de la porcion del plano complejo di"
  .ascii  "bujada, expresado en forma binomica. El valor por defect"
  .ascii  "o sera 0 + 0i  \n\000"
```

```
  .align  2
$LC28:
  .ascii  "\t %s -w --width, especifica el ancho del rectangulo que"
  .ascii  " contiene la region del plano complejo que estamos por d"
  .ascii  "ibujar. El valor por defecto sera 4 \n\000"
  .align  2
$LC29:
  .ascii  "\t %s -H --height, sirve, en forma similar, para especif"
  .ascii  "icar el alto del rectangulo a dibujar. El valor por defe"
  .ascii  "cto sera 4 \n\000"
  .align  2
$LC30:
  .ascii  "\t %s -o --output, permite colocar la imagen de salida, "
  .ascii  "(en formato PGM) en el archivo pasado como argumento; o "
  .ascii  "por salida estandar -cout- si el argumento es \"-\" \n\000"
  .align  2
$LC31:
  .ascii  "Examples: \n\000"
  .align  2
$LC32:
  .ascii  "\t %s -o uno.pgm \n\000"
  .align  2
$LC33:
  .ascii  "\t %s -c +0.282-0.01i -w 0.005 -H 0.005 -o dos.pgm \n\000"
  .align  2
$LC34:
  .ascii  "\t %s -r 1x1 -o - \n\000"
  .text
  .align  2
  .globl  usage
  .ent  usage
usage:
  .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
  sw  $4,40($fp)
  la  $4,$LC23
  la  $25,printf
  jal $31,$25
  la  $4,$LC24
  lw  $5,40($fp)
  la  $25,printf
  jal $31,$25
  la  $4,$LC25
  lw  $5,40($fp)
  la  $25,printf
  jal $31,$25
  la  $4,$LC26
  lw  $5,40($fp)
  la  $25,printf
  jal $31,$25
  la  $4,$LC27
  lw  $5,40($fp)
  la  $25,printf
  jal $31,$25
  la  $4,$LC28
  lw  $5,40($fp)
  la  $25,printf
  jal $31,$25
  la  $4,$LC29
```

```
  lw  $5,40($fp)
  la  $25,printf
  jal $31,$25
  la  $4,$LC30
  lw  $5,40($fp)
  la  $25,printf
  jal $31,$25
  la  $4,$LC31
  la  $25,printf
  jal $31,$25
  la  $4,$LC32
  lw  $5,40($fp)
  la  $25,printf
  jal $31,$25
  la  $4,$LC33
  lw  $5,40($fp)
  la  $25,printf
  jal $31,$25
  la  $4,$LC34
  lw  $5,40($fp)
  la  $25,printf
  jal $31,$25
  move  $sp,$fp
  lw  $31,32($sp)
  lw  $fp,28($sp)
  addu  $sp,$sp,40
  j $31
  .end  usage
  .size usage, .-usage
  .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```