Trabajo Prático 0

Agustin Alejandro Linari, *Padrón Nro. 81.783*
agustinlinari@gmail.com

Juan Ignacio López Pecora, *Padrón Nro. 84.700*
jlopezpecora@gmail.com

Pablo Daniel Sívori, *Padrón: 84.026*
sivoridaniel@gmail.com

2° Cuatrimestre de 2016
66.20 Organizacion de Computadoras
Facultad de Ingenieria, Universidad de Buenos Aires

13 de septiembre de 2016

Resumen

En presente trabajo se diseña un programa en el lenguaje C que permite graficar el conjunto de Julia y sus vecindades para un polinomio cuadrático.

Índice

Parte I

Desarrollo

1. Introduccion

El objetivo del presente trabajo práctico es familiarizarse con las herramientas de software a utilizar durante el transcurso de la materia. Para ello implementaremos un programa en lenguaje C que dado ciertos parámetros de entrada se ocupará de generar una imagen en formato PGM del conjunto de Julia de un polinómio cuadrático. Finalmente compilaremos el programa en el emulador GXemul para poder obtener el código de instrucciones Mips32.

2. Build

El correspondiente informe se puede construir utilizando el make. Luego se pueden ejecutar los tests corriendo el script run-tests.sh.

```
$ make
$ ./run-tests.sh
```

3. Diseño e Implementación del Programa

Para el desarrollo del programa se utilizó como base el pseudocódigo proporcionado en el enunciado del trabajo práctico. B Para su codificación en el lenguaje C, se necesitó crear un TDA complex para encapsular las operaciones con números complejos. Mediante la utilización de la libreria math, se desarrollaron funciones para las operaciones requeridas en el cómputo del algoritmo generador del fractal. Es decir, se implementaron la multiplicación, adición y módulo de complejos. También se agregó una función adicional que permite mapear cada uno de los pixels al punto del plano complejo correspondiente.

El primer paso es dividir el plano complejo según la resolución.

El ancho de cada división es el ancho del plano (w) dividido la resolución horizontal (x).

Δw = w/x

El alto de cada división es el alto del plano (h) dividido la resolución vertical (y).

Δh = h/y

El punto superior izquierdo del plano (z_0), se obtiene a partir del centro del plano (c).

$\mathrm{Re}(z_0) = \mathrm{Re}(c) - w/2$

$\mathrm{Im}(z_0) = \mathrm{Im}(c) + h/2$

Luego dado un pixel (i, j) se obtiene el z correspondiente.

$\mathrm{Re}(z) = \mathrm{Re}(z_0) + \Delta w/2 + \Delta w \cdot i$

$\mathrm{Im}(z) = \mathrm{Im}(z_0) - \Delta h/2 - \Delta h \cdot j$

En cuanto al formato de salida para la imagen, se utilizó el formato Plain PGM.

Los argumentos que llegan al programa se parsearon utilizando la librería getopt. El código fuente del programa se puede encontrar en el anexo A.

4. Corridas de Programa

4.1. Ejemplos

Primero realizamos la prueba de generar la imagen que se muestra de ejemplo en el enunciado. Para ello generamos la imagen con los valores dados por defecto. La imagen resultante se puede ver en la carpeta test con el nombre uno.pgm.

```
$ ./tp0 -o uno.pgm
```

Para el siguiente caso utilizaremos el parámetro C con el valor +0,285 - 0i, manteniendo el resto de los valores por defecto. Esta figura se puede ver en la carpeta test con el nombre dos.pgm.

```
$ ./tp0 -C 0.285+0i -o dos.pgm
```

En el tercer caso, generaremos la imagen utilizando C con el valor -0.8 + 0.156i. El resultado se puede ver en la imagen tres.pgm.

```
$ ./tp0 -C -0.8+0.156i -o tres.pgm
```

El cuarto caso, utilizaremos C con el valor 0.8i. El resultado se muestra en la imagen cuatro.pgm.

```
$ ./tp0 -C 0+0.8i -o cuatro.pgm
```

Por último, mostramos el tercer caso pero haciendo un zoom en el centro.

```
$ ./tp0 -C -0.8+0.156i -w 0.5 -H 0.5 -o cinco.pgm
```

4.2. Pruebas

Luego de verificar que el programa genera las imagenes correspondiente a los ejemplos, procedimos a ejecutar los casos de prueba. Para ello nos posicionamos en la carpeta raíz que contiene al archivo run-test.sh y luego realizamos la ejecución de este archivo haciendo sh run-test.sh, retornando como output los resultados correspondientes a cada prueba. A continuación detallamos los resultados de las pruebas realizadas.

1. **Generación de una imagen de 1 punto de lado, centrada en el origen del plano complejo.**

   Este test prueba el caso de un sólo punto cuyo brillo vale 19.

   Para esto se uso el valor esperado expected1.pgm que se encuentra en la carpeta test.

2. **Un punto que no pertenece al conjunto.**

   En este caso, se intenta probar que el punto ingresado no pertenece al conjunto de Julia, por lo tanto el valor de su brillo debe ser 0. Para esto se uso el valor esperado expected2.pgm que se encuentra en la carpeta test.

3. **Imagen imposible.**

   En este caso de prueba, se intenta ingresar una imagen de ancho 0, con lo cual se espera que el programa retorne un mensaje de error. Para esto se uso el valor esperado expected3 que se encuentra en la carpeta test.

4. **Archivo de salida imposible.**

   Se prueba el caso de ingresar un directorio en vez de un archivo como output.

   Para esto se uso el valor esperado expected4 que se encuentra en la carpeta test.

5. **Coordenadas complejas imposibles.**

   Se ingresan valores que no corresponden a un número complejo y se busca comparar el mensaje de error retornado. Para esto se uso el valor esperado expected5 que se encuentra en la carpeta test.

6. **Argumentos de líneas de compando vacíos.**

   En el último caso, se ingresa un argumento sin valor y el programa deberá emitir un mensaje de error mostrando el uso correcto de los argumentos. Para esto se uso el valor esperado expected6 que se encuentra en la carpeta test.

Parte II

Apendice

A. Codigo fuente

../src/complex.h

```
typedef struct complex {
    double re;
    double im;
} complex_t;

void complex_create(complex_t* z, double re, double im);

void complex_destroy(complex_t* z);

void complex_add(complex_t* z, complex_t w);

void complex_mult(complex_t* z, complex_t w);

double complex_abs(complex_t* z);

void complex_map(complex_t* z, int i, int j, double dw, double dh,
    double re0, double im0);
```

../src/complex.c

```
#include "complex.h"
#include <math.h>


void complex_create(complex_t* z, double re, double im) {
    z->re = re;
    z->im = im;
}

void complex_destroy(complex_t* z) {}

void complex_add(complex_t* z, complex_t w) {
    z->re = z->re + w.re;
    z->im = z->im + w.im;
}

void complex_mult(complex_t* z, complex_t w) {
    double re = z->re;
    double im = z->im;
    z->re = (re*w.re)-(im*w.im);
    z->im = (re*w.im)+(im*w.re);
}

double complex_abs(complex_t* z) {
   double abs =sqrt((z->re*z->re)+(z->im*z->im));
   return abs;
```

```
}

void complex_map(complex_t* z, int i, int j, double dw, double dh,
    double re0, double im0){
    double re = re0+dw/2+dw*i;
    double im = im0-dh/2-dh*j;
    z->re = re;
    z->im = im;
}
```

../src/app.c

```
#include <ctype.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include "complex.h"

#define VERSION 1.0

#define Z_RE 0
#define Z_IM 0

#define C_RE 0.285
#define C_IM -0.01

#define RESOLUTION_X 640
#define RESOLUTION_Y 480

#define WIDTH 4
#define HEIGHT 4

#define N_MAX 255

#define ERR_ARG 1

static void print_help();
static void print_version();
static void print_invalid_arg(int opt, char* arg);
static int set_resolution(char* res, int* rx, int* ry);
static int set_size(char* val, double* dim);
static int set_complex(char* val, complex_t* z);
static void drawJulia(int x, int y, double w, double h, complex_t
    zc, complex_t c,int N, FILE* file);

const char* short_options = "hvr:c:C:w:H:o:";

static struct option options[] = {
    { "help", no_argument, NULL, 'h' },
    { "version", no_argument, NULL, 'v' },
    { "resolution", required_argument, NULL, 'r' },
    { "center", required_argument, NULL, 'c' },
    { "param", required_argument, NULL, 'C' },
    { "width", required_argument, NULL, 'w' },
    { "height", required_argument, NULL, 'H' },
```

```
45      { "output", required_argument, NULL, 'o' },
46      { 0, 0 }
47  };
48
49  int main(int argc, char** argv) {
50      FILE* file = stdout;
51
52      complex_t zc;
53      zc.re = Z_RE;
54      zc.im = Z_IM;
55
56      complex_t c;
57      c.re = C_RE;
58      c.im = C_IM;
59
60      int rx = RESOLUTION_X;
61      int ry = RESOLUTION_Y;
62
63      double w = WIDTH;
64      double h = HEIGHT;
65
66          int param = 0;
67          int index = 0;
68          while ((param = getopt_long(argc, argv, short_options,
                options, &index)) != -1) {
69              switch (param) {
70                      case 'h':
71                              print_help();
72                              return EXIT_SUCCESS;
73                      case 'v':
74                              print_version();
75                              return EXIT_SUCCESS;
76                  case 'r':
77                              if (set_resolution(optarg, &rx, &ry
                                    ) != 0) {
78                  print_invalid_arg(param, optarg);
79                  print_help();
80                              return ERR_ARG;
81                              }
82                              break;
83                  case 'c':
84                              if (set_complex(optarg, &zc) != 0)
                                    {
85                  print_invalid_arg(param, optarg);
86                  print_help();
87                              return ERR_ARG;
88                              }
89                              break;
90                  case 'C':
91                              if (set_complex(optarg, &c) != 0) {
92                  print_invalid_arg(param, optarg);
93                  print_help();
94                              return ERR_ARG;
95                              }
96                              break;
97              case 'w':
98                  if (set_size(optarg, &w) != 0) {
```

```
99                      print_invalid_arg(param, optarg);
100                     print_help();
101                                 return ERR_ARG;
102                             }
103                             break;
104             case 'H':
105                 if (set_size(optarg, &h) != 0) {
106                     print_invalid_arg(param, optarg);
107                     print_help();
108                                 return ERR_ARG;
109                             }
110                             break;
111             case 'o':
112                 if (strlen(optarg) > 0 && optarg[0] == '-') {
113                     break;
114                 }
115                 errno = 0;
116                 if ((file = fopen(optarg,"w")) == NULL) {
117                     perror("Error on fopen");
118                     return ERR_ARG;
119                 }
120                             break;
121             case '?':
122                             print_help();
123                             return ERR_ARG;
124                         default:
125                             print_help();
126                             return EXIT_SUCCESS;
127                 }
128         }
129     drawJulia(rx, ry, w, h, zc, c,N_MAX, file);
130     fclose(file);
131     return EXIT_SUCCESS;
132 }
133
134 static void print_version() {
135     printf("Version: %-1.1f\n", VERSION);
136 }
137
138 static void print_help() {
139         printf("Usage:\n"
140            "  tp0 -h\n"
141            "  tp0 -v\n"
142            "  tp0 -r <axb> -c <> -C <> -w <width> -H <height> -o <
                 filepath>\n"
143            "  tp0 < in_file > out_file\n"
144            "Options:\n"
145            "  -r                Set resolution. [640x480] by default
                 .\n"
146            "  -c                Set image center. [0+0i] by default
                 .\n"
147            "  -C                Set c value. [0.285-0.01i] by
                 default.\n"
148            "  -w                Set width. [4] by default.\n"
149            "  -H                Set height. [4] by default.\n"
150            "  -o                Set output file. [std] by default.\n
                 "
```

```
151             "  -h, --help        Print this information and quit.\n"
152             "Examples:\n"
153             "  tp0 -r 1024x800 -w 24 -H 16 -o out.pgm\n"
154             "  cat in.txt | tp0 > out.txt\n");
155 }
156
157 static void print_invalid_arg(int opt, char* arg) {
158         printf("Invalid argument <%s> for the option <%c>\n", arg,
                opt);
159 }
160
161 static int set_resolution(char* res, int* rx, int* ry) {
162     int i = 0;
163     int x = 0;
164     int y = 0;
165     while (res[i] != 'x' && res[i] != 'X' && res[i] != '\0') {
166         if (!isdigit(res[i])) {
167             return -1;
168         }
169         x = x *10 + (res[i] - '0');
170         ++i;
171     }
172     if (x == 0 || i == strlen(res)) {
173         return -1;
174     }
175     i++;
176     while (res[i] != '\0') {
177         if (!isdigit(res[i])) {
178             return -1;
179         }
180         y = y *10 + (res[i] - '0');
181         ++i;
182     }
183     if (y == 0) {
184         return -1;
185     }
186     *rx = x;
187     *ry = y;
188     return 0;
189 }
190
191 static int set_complex(char* val, complex_t* z) {
192     if (strlen(val) == 0) {
193         printf("A\n");
194         return -1;
195     }
196
197     int i = 0;
198     double re = 0;
199     double im = 0;
200
201     //RE(z)
202     for (i = 1; val[i] != '+' && val[i] != '-' && val[i] != '\0';
            ++i) {
203         ;
204     }
205     //copy RE(z) to a new string
```

```
206     char* re_str = malloc(i+1);
207     int j;
208     for (j = 0; j < i; ++j) {
209         re_str[j] = val[j];
210     }
211     re_str[j] = '\0';
212 
213     char *endptr;
214     errno = 0;    // To distinguish success/failure after call
215     re = strtod(re_str, &endptr);
216     if (errno == ERANGE) {  //check overflow or underflow
217         perror("out of range");
218         return -1;
219     }
220     if (*endptr != '\0') {  //test for invalid characters
221         printf("Re value not a number: <%s>\n", endptr);
222         free(re_str);
223         return -1;
224     }
225     free(re_str);
226 
227     //IM(z)
228     char* im_str = val + i; //define img(z) string
229 
230     //test for character 'i' presence
231     if (strlen(im_str) > 0 && im_str[strlen(im_str) - 1] != 'i') {
232         printf("Complex number must follow the a+bi notation.\n");
233         return -1;
234     }
235     im = strtod(im_str, &endptr);
236     //test for invalid characters
237     if (*endptr != 'i') {
238         return -1;
239     }
240 
241     z->re = re;
242     z->im = im;
243     return 0;
244 }
245 
246 static int set_size(char* val, double* dim) {
247     char *endptr;
248     errno = 0;    // To distinguish success/failure after call
249     *dim = strtod(val, &endptr);
250     if (errno == ERANGE) {  //check overflow or underflow
251         perror("out of range");
252         return -1;
253     }
254     if (*endptr != '\0') {  //test for invalid characters
255         printf("set_size val argument not a number: <%s>\n", endptr
                );
256         return -1;
257     }
258 
259     return 0;
260 }
261 
```

```
262 static void drawJulia(int x, int y, double w, double h, complex_t
        zc, complex_t c,int N, FILE* file){
263     double dw=w/x;
264     double dh = h/y;
265     double re0 = zc.re - w/2;
266     double im0 = zc.im + h/2;
267
268     fprintf(file, "P2\n");
269     fprintf(file, "%d\n", x);
270     fprintf(file, "%d\n", y);
271     fprintf(file, "%d\n", N);
272
273     for (int j=0; j<y; ++j) {
274         for (int i =0; i<x; ++i) {
275             complex_t z;
276             complex_map(&z, i, j, dw, dh, re0, im0);
277             int brillo = N;
278
279             for (int k=0; k<N-1;++k) {
280                 if (complex_abs(&z)>2) {
281                     brillo = k;
282                     break;
283                 }
284                 complex_mult(&z,z);
285                 complex_add(&z,c);
286             }
287             fprintf(file,"%d ",brillo);
288         }
289             fprintf(file,"\n");
290         }
291 }
```

```
    .file   1 "src/complex.c"
    .section .mdebug.abi32
    .previous
    .abicalls
    .text
    .align  2
    .globl  complex_create
    .ent    complex_create
complex_create:
    .frame  $fp,16,$ra      # vars= 0, regs= 2/0, args= 0, extra= 8
    .mask   0x50000000,-4
    .fmask  0x00000000,0
    .set    noreorder
    .cpload $t9
    .set    reorder
    subu    $sp,$sp,16
    .cprestore 0
    sw  $fp,12($sp)
    sw  $gp,8($sp)
    move    $fp,$sp
    sw  $a0,16($fp)
    sw  $a2,24($fp)
    sw  $a3,28($fp)
    lw  $v0,16($fp)
    l.d $f0,24($fp)
    s.d $f0,0($v0)
    lw  $v0,16($fp)
    l.d $f0,32($fp)
    s.d $f0,8($v0)
    move    $sp,$fp
    lw  $fp,12($sp)
    addu    $sp,$sp,16
    j   $ra
    .end    complex_create
    .size   complex_create, .-complex_create
    .align  2
    .globl  complex_destroy
    .ent    complex_destroy
complex_destroy:
    .frame  $fp,16,$ra      # vars= 0, regs= 2/0, args= 0, extra= 8
    .mask   0x50000000,-4
    .fmask  0x00000000,0
    .set    noreorder
    .cpload $t9
    .set    reorder
    subu    $sp,$sp,16
    .cprestore 0
    sw  $fp,12($sp)
    sw  $gp,8($sp)
    move    $fp,$sp
    sw  $a0,16($fp)
    move    $sp,$fp
    lw  $fp,12($sp)
    addu    $sp,$sp,16
    j   $ra
    .end    complex_destroy
    .size   complex_destroy, .-complex_destroy
```

```
58      .align  2
59      .globl  complex_add
60      .ent    complex_add
61  complex_add:
62      .frame  $fp,16,$ra      # vars= 0, regs= 2/0, args= 0, extra= 8
63      .mask   0x50000000,-4
64      .fmask  0x00000000,0
65      .set    noreorder
66      .cpload $t9
67      .set    reorder
68      subu    $sp,$sp,16
69      .cprestore 0
70      sw  $fp,12($sp)
71      sw  $gp,8($sp)
72      move    $fp,$sp
73      sw  $a0,16($fp)
74      sw  $a2,24($fp)
75      sw  $a3,28($fp)
76      lw  $v1,16($fp)
77      lw  $v0,16($fp)
78      l.d $f2,0($v0)
79      l.d $f0,24($fp)
80      add.d   $f0,$f2,$f0
81      s.d $f0,0($v1)
82      lw  $v1,16($fp)
83      lw  $v0,16($fp)
84      l.d $f2,8($v0)
85      l.d $f0,32($fp)
86      add.d   $f0,$f2,$f0
87      s.d $f0,8($v1)
88      move    $sp,$fp
89      lw  $fp,12($sp)
90      addu    $sp,$sp,16
91      j   $ra
92      .end    complex_add
93      .size   complex_add, .-complex_add
94      .align  2
95      .globl  complex_mult
96      .ent    complex_mult
97  complex_mult:
98      .frame  $fp,32,$ra      # vars= 16, regs= 2/0, args= 0, extra=
            8
99      .mask   0x50000000,-4
100     .fmask  0x00000000,0
101     .set    noreorder
102     .cpload $t9
103     .set    reorder
104     subu    $sp,$sp,32
105     .cprestore 0
106     sw  $fp,28($sp)
107     sw  $gp,24($sp)
108     move    $fp,$sp
109     sw  $a0,32($fp)
110     sw  $a2,40($fp)
111     sw  $a3,44($fp)
112     lw  $v0,32($fp)
113     l.d $f0,0($v0)
```

```
114     s.d $f0,8($fp)
115     lw  $v0,32($fp)
116     l.d $f0,8($v0)
117     s.d $f0,16($fp)
118     lw  $v0,32($fp)
119     l.d $f2,8($fp)
120     l.d $f0,40($fp)
121     mul.d   $f4,$f2,$f0
122     l.d $f2,16($fp)
123     l.d $f0,48($fp)
124     mul.d   $f0,$f2,$f0
125     sub.d   $f0,$f4,$f0
126     s.d $f0,0($v0)
127     lw  $v0,32($fp)
128     l.d $f2,8($fp)
129     l.d $f0,48($fp)
130     mul.d   $f4,$f2,$f0
131     l.d $f2,16($fp)
132     l.d $f0,40($fp)
133     mul.d   $f0,$f2,$f0
134     add.d   $f0,$f4,$f0
135     s.d $f0,8($v0)
136     move    $sp,$fp
137     lw  $fp,28($sp)
138     addu    $sp,$sp,32
139     j   $ra
140     .end    complex_mult
141     .size   complex_mult, .-complex_mult
142     .align  2
143     .globl  complex_abs
144     .ent    complex_abs
145 complex_abs:
146     .frame  $fp,48,$ra      # vars= 8, regs= 3/0, args= 16, extra=
            8
147     .mask   0xd0000000,-8
148     .fmask  0x00000000,0
149     .set    noreorder
150     .cpload $t9
151     .set    reorder
152     subu    $sp,$sp,48
153     .cprestore 16
154     sw  $ra,40($sp)
155     sw  $fp,36($sp)
156     sw  $gp,32($sp)
157     move    $fp,$sp
158     sw  $a0,48($fp)
159     lw  $v0,48($fp)
160     lw  $v1,48($fp)
161     l.d $f2,0($v0)
162     l.d $f0,0($v1)
163     mul.d   $f4,$f2,$f0
164     lw  $v0,48($fp)
165     lw  $v1,48($fp)
166     l.d $f2,8($v0)
167     l.d $f0,8($v1)
168     mul.d   $f0,$f2,$f0
169     add.d   $f0,$f4,$f0
```

```
170     mov.d   $f12,$f0
171     la  $t9,sqrt
172     jal $ra,$t9
173     s.d $f0,24($fp)
174     l.d $f0,24($fp)
175     move    $sp,$fp
176     lw  $ra,40($sp)
177     lw  $fp,36($sp)
178     addu    $sp,$sp,48
179     j   $ra
180     .end    complex_abs
181     .size   complex_abs, .-complex_abs
182     .rdata
183     .align  3
184 $LC0:
185     .word   0
186     .word   1073741824
187     .text
188     .align  2
189     .globl  complex_map
190     .ent    complex_map
191 complex_map:
192     .frame  $fp,32,$ra      # vars= 16, regs= 2/0, args= 0, extra=
            8
193     .mask   0x50000000,-4
194     .fmask  0x00000000,0
195     .set    noreorder
196     .cpload $t9
197     .set    reorder
198     subu    $sp,$sp,32
199     .cprestore 0
200     sw  $fp,28($sp)
201     sw  $gp,24($sp)
202     move    $fp,$sp
203     sw  $a0,32($fp)
204     sw  $a1,36($fp)
205     sw  $a2,40($fp)
206     l.d $f2,48($fp)
207     l.d $f0,$LC0
208     div.d   $f2,$f2,$f0
209     l.d $f0,64($fp)
210     add.d   $f4,$f2,$f0
211     l.s $f0,36($fp)
212     cvt.d.w $f2,$f0
213     l.d $f0,48($fp)
214     mul.d   $f0,$f2,$f0
215     add.d   $f0,$f4,$f0
216     s.d $f0,8($fp)
217     l.d $f2,56($fp)
218     l.d $f0,$LC0
219     div.d   $f2,$f2,$f0
220     l.d $f0,72($fp)
221     sub.d   $f4,$f0,$f2
222     l.s $f0,40($fp)
223     cvt.d.w $f2,$f0
224     l.d $f0,56($fp)
225     mul.d   $f0,$f2,$f0
```

```
226     sub.d   $f0,$f4,$f0
227     s.d $f0,16($fp)
228     lw  $v0,32($fp)
229     l.d $f0,8($fp)
230     s.d $f0,0($v0)
231     lw  $v0,32($fp)
232     l.d $f0,16($fp)
233     s.d $f0,8($v0)
234     move    $sp,$fp
235     lw  $fp,28($sp)
236     addu    $sp,$sp,32
237     j   $ra
238     .end    complex_map
239     .size   complex_map, .-complex_map
240     .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

../MIPS32/complex.s

```
1       .file   1 "src/app.c"
2       .section .mdebug.abi32
3       .previous
4       .abicalls
5       .rdata
6       .align  2
7   $LC0:
8       .ascii  "hvr:c:C:w:H:o:\000"
9       .globl  short_options
10      .data
11      .align  2
12      .type   short_options, @object
13      .size   short_options, 4
14  short_options:
15      .word   $LC0
16      .rdata
17      .align  2
18  $LC1:
19      .ascii  "help\000"
20      .align  2
21  $LC2:
22      .ascii  "version\000"
23      .align  2
24  $LC3:
25      .ascii  "resolution\000"
26      .align  2
27  $LC4:
28      .ascii  "center\000"
29      .align  2
30  $LC5:
31      .ascii  "param\000"
32      .align  2
33  $LC6:
34      .ascii  "width\000"
35      .align  2
36  $LC7:
37      .ascii  "height\000"
38      .align  2
39  $LC8:
```

```
40      .ascii  "output\000"
41      .data
42      .align  2
43      .type   options, @object
44      .size   options, 144
45  options:
46      .word   $LC1
47      .word   0
48      .word   0
49      .word   104
50      .word   $LC2
51      .word   0
52      .word   0
53      .word   118
54      .word   $LC3
55      .word   1
56      .word   0
57      .word   114
58      .word   $LC4
59      .word   1
60      .word   0
61      .word   99
62      .word   $LC5
63      .word   1
64      .word   0
65      .word   67
66      .word   $LC6
67      .word   1
68      .word   0
69      .word   119
70      .word   $LC7
71      .word   1
72      .word   0
73      .word   72
74      .word   $LC8
75      .word   1
76      .word   0
77      .word   111
78      .word   0
79      .word   0
80      .space  8
81      .rdata
82      .align  2
83  $LC12:
84      .ascii  "w\000"
85      .align  2
86  $LC13:
87      .ascii  "Error on fopen\000"
88      .align  3
89  $LC9:
90      .word   -1546188227
91      .word   1070742896
92      .align  3
93  $LC10:
94      .word   1202590843
95      .word   -1081836831
96      .align  3
```

```
97  $LC11:
98      .word   0
99      .word   1074790400
100     .text
101     .align  2
102     .globl  main
103     .ent    main
104 main:
105     .frame  $fp,168,$ra     # vars= 80, regs= 3/0, args= 64, extra=
            8
106     .mask   0xd0000000,-8
107     .fmask  0x00000000,0
108     .set    noreorder
109     .cpload $t9
110     .set    reorder
111     subu    $sp,$sp,168
112     .cprestore 64
113     sw  $ra,160($sp)
114     sw  $fp,156($sp)
115     sw  $gp,152($sp)
116     move    $fp,$sp
117     sw  $a0,168($fp)
118     sw  $a1,172($fp)
119     la  $v0,__sF+88
120     sw  $v0,72($fp)
121     sw  $zero,80($fp)
122     sw  $zero,84($fp)
123     sw  $zero,88($fp)
124     sw  $zero,92($fp)
125     l.d $f0,$LC9
126     s.d $f0,96($fp)
127     l.d $f0,$LC10
128     s.d $f0,104($fp)
129     li  $v0,640         # 0x280
130     sw  $v0,112($fp)
131     li  $v0,480         # 0x1e0
132     sw  $v0,116($fp)
133     l.d $f0,$LC11
134     s.d $f0,120($fp)
135     l.d $f0,$LC11
136     s.d $f0,128($fp)
137     sw  $zero,136($fp)
138     sw  $zero,140($fp)
139 $L18:
140     addu    $v0,$fp,140
141     sw  $v0,16($sp)
142     lw  $a0,168($fp)
143     lw  $a1,172($fp)
144     lw  $a2,short_options
145     la  $a3,options
146     la  $t9,getopt_long
147     jal $ra,$t9
148     sw  $v0,136($fp)
149     lw  $v1,136($fp)
150     li  $v0,-1          # 0xffffffffffffffff
151     bne $v1,$v0,$L20
152     b   $L19
```

```
153 $L20:
154     lw   $v0,136($fp)
155     addu     $v0,$v0,-63
156     sw   $v0,148($fp)
157     lw   $v1,148($fp)
158     sltu     $v0,$v1,57
159     beq  $v0,$zero,$L38
160     lw   $v0,148($fp)
161     sll  $v1,$v0,2
162     la   $v0,$L39
163     addu     $v0,$v1,$v0
164     lw   $v0,0($v0)
165     .cpadd   $v0
166     j    $v0
167     .rdata
168     .align   2
169 $L39:
170     .gpword  $L37
171     .gpword  $L38
172     .gpword  $L38
173     .gpword  $L38
174     .gpword  $L28
175     .gpword  $L38
176     .gpword  $L38
177     .gpword  $L38
178     .gpword  $L38
179     .gpword  $L32
180     .gpword  $L38
181     .gpword  $L38
182     .gpword  $L38
183     .gpword  $L38
184     .gpword  $L38
185     .gpword  $L38
186     .gpword  $L38
187     .gpword  $L38
188     .gpword  $L38
189     .gpword  $L38
190     .gpword  $L38
191     .gpword  $L38
192     .gpword  $L38
193     .gpword  $L38
194     .gpword  $L38
195     .gpword  $L38
196     .gpword  $L38
197     .gpword  $L38
198     .gpword  $L38
199     .gpword  $L38
200     .gpword  $L38
201     .gpword  $L38
202     .gpword  $L38
203     .gpword  $L38
204     .gpword  $L38
205     .gpword  $L38
206     .gpword  $L26
207     .gpword  $L38
208     .gpword  $L38
209     .gpword  $L38
```

```
210     .gpword $L38
211     .gpword $L22
212     .gpword $L38
213     .gpword $L38
214     .gpword $L38
215     .gpword $L38
216     .gpword $L38
217     .gpword $L38
218     .gpword $L34
219     .gpword $L38
220     .gpword $L38
221     .gpword $L24
222     .gpword $L38
223     .gpword $L38
224     .gpword $L38
225     .gpword $L23
226     .gpword $L30
227     .text
228 $L22:
229     la  $t9,print_help
230     jal $ra,$t9
231     sw  $zero,144($fp)
232     b   $L17
233 $L23:
234     la  $t9,print_version
235     jal $ra,$t9
236     sw  $zero,144($fp)
237     b   $L17
238 $L24:
239     addu    $v0,$fp,112
240     addu    $v1,$fp,116
241     lw  $a0,optarg
242     move    $a1,$v0
243     move    $a2,$v1
244     la  $t9,set_resolution
245     jal $ra,$t9
246     beq $v0,$zero,$L18
247     lw  $a0,136($fp)
248     lw  $a1,optarg
249     la  $t9,print_invalid_arg
250     jal $ra,$t9
251     la  $t9,print_help
252     jal $ra,$t9
253     li  $v0,1           # 0x1
254     sw  $v0,144($fp)
255     b   $L17
256 $L26:
257     addu    $v0,$fp,80
258     lw  $a0,optarg
259     move    $a1,$v0
260     la  $t9,set_complex
261     jal $ra,$t9
262     beq $v0,$zero,$L18
263     lw  $a0,136($fp)
264     lw  $a1,optarg
265     la  $t9,print_invalid_arg
266     jal $ra,$t9
```

```
267     la   $t9,print_help
268     jal  $ra,$t9
269     li   $v1,1                # 0x1
270     sw   $v1,144($fp)
271     b    $L17
272 $L28:
273     addu     $v0,$fp,96
274     lw   $a0,optarg
275     move     $a1,$v0
276     la   $t9,set_complex
277     jal  $ra,$t9
278     beq  $v0,$zero,$L18
279     lw   $a0,136($fp)
280     lw   $a1,optarg
281     la   $t9,print_invalid_arg
282     jal  $ra,$t9
283     la   $t9,print_help
284     jal  $ra,$t9
285     li   $v0,1                # 0x1
286     sw   $v0,144($fp)
287     b    $L17
288 $L30:
289     addu     $v0,$fp,120
290     lw   $a0,optarg
291     move     $a1,$v0
292     la   $t9,set_size
293     jal  $ra,$t9
294     beq  $v0,$zero,$L18
295     lw   $a0,136($fp)
296     lw   $a1,optarg
297     la   $t9,print_invalid_arg
298     jal  $ra,$t9
299     la   $t9,print_help
300     jal  $ra,$t9
301     li   $v1,1                # 0x1
302     sw   $v1,144($fp)
303     b    $L17
304 $L32:
305     addu     $v0,$fp,128
306     lw   $a0,optarg
307     move     $a1,$v0
308     la   $t9,set_size
309     jal  $ra,$t9
310     beq  $v0,$zero,$L18
311     lw   $a0,136($fp)
312     lw   $a1,optarg
313     la   $t9,print_invalid_arg
314     jal  $ra,$t9
315     la   $t9,print_help
316     jal  $ra,$t9
317     li   $v0,1                # 0x1
318     sw   $v0,144($fp)
319     b    $L17
320 $L34:
321     lw   $v0,optarg
322     lb   $v0,0($v0)
323     beq  $v0,$zero,$L35
```

```
324     lw   $v0,optarg
325     lb   $v1,0($v0)
326     li   $v0,45              # 0x2d
327     bne  $v1,$v0,$L35
328     b    $L18
329 $L35:
330     la   $t9,__errno
331     jal  $ra,$t9
332     sw   $zero,0($v0)
333     lw   $a0,optarg
334     la   $a1,$LC12
335     la   $t9,fopen
336     jal  $ra,$t9
337     sw   $v0,72($fp)
338     lw   $v0,72($fp)
339     bne  $v0,$zero,$L18
340     la   $a0,$LC13
341     la   $t9,perror
342     jal  $ra,$t9
343     li   $v1,1               # 0x1
344     sw   $v1,144($fp)
345     b    $L17
346 $L37:
347     la   $t9,print_help
348     jal  $ra,$t9
349     li   $v0,1               # 0x1
350     sw   $v0,144($fp)
351     b    $L17
352 $L38:
353     la   $t9,print_help
354     jal  $ra,$t9
355     sw   $zero,144($fp)
356     b    $L17
357 $L19:
358     l.d  $f0,128($fp)
359     s.d  $f0,16($sp)
360     lw   $v0,80($fp)
361     sw   $v0,24($sp)
362     lw   $v0,84($fp)
363     sw   $v0,28($sp)
364     lw   $v0,88($fp)
365     sw   $v0,32($sp)
366     lw   $v0,92($fp)
367     sw   $v0,36($sp)
368     lw   $v0,96($fp)
369     sw   $v0,40($sp)
370     lw   $v0,100($fp)
371     sw   $v0,44($sp)
372     lw   $v0,104($fp)
373     sw   $v0,48($sp)
374     lw   $v0,108($fp)
375     sw   $v0,52($sp)
376     li   $v0,255             # 0xff
377     sw   $v0,56($sp)
378     lw   $v0,72($fp)
379     sw   $v0,60($sp)
380     lw   $a0,112($fp)
```

```
381     lw   $a1,116($fp)
382     lw   $a2,120($fp)
383     lw   $a3,124($fp)
384     la   $t9,drawJulia
385     jal  $ra,$t9
386     lw   $a0,72($fp)
387     la   $t9,fclose
388     jal  $ra,$t9
389     sw   $zero,144($fp)
390 $L17:
391     lw   $v0,144($fp)
392     move     $sp,$fp
393     lw   $ra,160($sp)
394     lw   $fp,156($sp)
395     addu     $sp,$sp,168
396     j    $ra
397     .end     main
398     .size    main, .-main
399     .rdata
400     .align   2
401 $LC14:
402     .ascii   "Version: %-1.1f\n\000"
403     .align   3
404 $LC15:
405     .word    0
406     .word    1072693248
407     .text
408     .align   2
409     .ent     print_version
410 print_version:
411     .frame   $fp,40,$ra      # vars= 0, regs= 3/0, args= 16, extra=
            8
412     .mask    0xd0000000,-8
413     .fmask   0x00000000,0
414     .set     noreorder
415     .cpload  $t9
416     .set     reorder
417     subu     $sp,$sp,40
418     .cprestore 16
419     sw   $ra,32($sp)
420     sw   $fp,28($sp)
421     sw   $gp,24($sp)
422     move     $fp,$sp
423     l.d  $f0,$LC15
424     la   $a0,$LC14
425     mfc1     $a2,$f0
426     mfc1     $a3,$f1
427     la   $t9,printf
428     jal  $ra,$t9
429     move     $sp,$fp
430     lw   $ra,32($sp)
431     lw   $fp,28($sp)
432     addu     $sp,$sp,40
433     j    $ra
434     .end     print_version
435     .size    print_version, .-print_version
436     .rdata
```

```
437     .align  2
438 $LC16:
439     .ascii  "Usage:\n"
440     .ascii  "  tp0 -h\n"
441     .ascii  "  tp0 -v\n"
442     .ascii  "  tp0 -r <axb> -c <> -C <> -w <width> -H <height> -o <
            fi"
443     .ascii  "lepath>\n"
444     .ascii  "  tp0 < in_file > out_file\n"
445     .ascii  "Options:\n"
446     .ascii  "  -r               Set resolution. [640x480] by
            default."
447     .ascii  "\n"
448     .ascii  "  -c               Set image center. [0+0i] by default
            .\n"
449     .ascii  "  -C               Set c value. [0.285-0.01i] by
            default"
450     .ascii  ".\n"
451     .ascii  "  -w               Set width. [4] by default.\n"
452     .ascii  "  -H               Set height. [4] by default.\n"
453     .ascii  "  -o               Set output file. [std] by default.\
            n"
454     .ascii  "  -h, --help       Print this information and quit.\n"
455     .ascii  "Examples:\n"
456     .ascii  "  tp0 -r 1024x800 -w 24 -H 16 -o out.pgm\n"
457     .ascii  "  cat in.txt | tp0 > out.txt\n\000"
458     .text
459     .align  2
460     .ent    print_help
461 print_help:
462     .frame  $fp,40,$ra      # vars= 0, regs= 3/0, args= 16, extra=
            8
463     .mask   0xd0000000,-8
464     .fmask  0x00000000,0
465     .set    noreorder
466     .cpload $t9
467     .set    reorder
468     subu    $sp,$sp,40
469     .cprestore 16
470     sw  $ra,32($sp)
471     sw  $fp,28($sp)
472     sw  $gp,24($sp)
473     move    $fp,$sp
474     la  $a0,$LC16
475     la  $t9,printf
476     jal $ra,$t9
477     move    $sp,$fp
478     lw  $ra,32($sp)
479     lw  $fp,28($sp)
480     addu    $sp,$sp,40
481     j   $ra
482     .end    print_help
483     .size   print_help, .-print_help
484     .rdata
485     .align  2
486 $LC17:
487     .ascii  "Invalid argument <%s> for the option <%c>\n\000"
```

```
488     .text
489     .align  2
490     .ent    print_invalid_arg
491 print_invalid_arg:
492     .frame  $fp,40,$ra      # vars= 0, regs= 3/0, args= 16, extra=
            8
493     .mask   0xd0000000,-8
494     .fmask  0x00000000,0
495     .set    noreorder
496     .cpload $t9
497     .set    reorder
498     subu    $sp,$sp,40
499     .cprestore 16
500     sw  $ra,32($sp)
501     sw  $fp,28($sp)
502     sw  $gp,24($sp)
503     move    $fp,$sp
504     sw  $a0,40($fp)
505     sw  $a1,44($fp)
506     la  $a0,$LC17
507     lw  $a1,44($fp)
508     lw  $a2,40($fp)
509     la  $t9,printf
510     jal $ra,$t9
511     move    $sp,$fp
512     lw  $ra,32($sp)
513     lw  $fp,28($sp)
514     addu    $sp,$sp,40
515     j   $ra
516     .end    print_invalid_arg
517     .size   print_invalid_arg, .-print_invalid_arg
518     .align  2
519     .ent    set_resolution
520 set_resolution:
521     .frame  $fp,56,$ra      # vars= 16, regs= 3/0, args= 16, extra=
            8
522     .mask   0xd0000000,-8
523     .fmask  0x00000000,0
524     .set    noreorder
525     .cpload $t9
526     .set    reorder
527     subu    $sp,$sp,56
528     .cprestore 16
529     sw  $ra,48($sp)
530     sw  $fp,44($sp)
531     sw  $gp,40($sp)
532     move    $fp,$sp
533     sw  $a0,56($fp)
534     sw  $a1,60($fp)
535     sw  $a2,64($fp)
536     sw  $zero,24($fp)
537     sw  $zero,28($fp)
538     sw  $zero,32($fp)
539 $L44:
540     lw  $v1,56($fp)
541     lw  $v0,24($fp)
542     addu    $v0,$v1,$v0
```

```
543     lb   $v1,0($v0)
544     li   $v0,120            # 0x78
545     beq  $v1,$v0,$L45
546     lw   $v1,56($fp)
547     lw   $v0,24($fp)
548     addu     $v0,$v1,$v0
549     lb   $v1,0($v0)
550     li   $v0,88             # 0x58
551     beq  $v1,$v0,$L45
552     lw   $v1,56($fp)
553     lw   $v0,24($fp)
554     addu     $v0,$v1,$v0
555     lb   $v0,0($v0)
556     bne  $v0,$zero,$L46
557     b    $L45
558 $L46:
559     lw   $v1,56($fp)
560     lw   $v0,24($fp)
561     addu     $v0,$v1,$v0
562     lb   $v1,0($v0)
563     lw   $v0,_ctype_
564     addu     $v0,$v1,$v0
565     addu     $v0,$v0,1
566     lbu  $v0,0($v0)
567     srl  $v0,$v0,2
568     andi     $v0,$v0,0x1
569     bne  $v0,$zero,$L48
570     li   $v0,-1             # 0xffffffffffffffff
571     sw   $v0,36($fp)
572     b    $L43
573 $L48:
574     lw   $v1,28($fp)
575     move     $v0,$v1
576     sll  $v0,$v0,2
577     addu     $v0,$v0,$v1
578     sll  $a0,$v0,1
579     lw   $v1,56($fp)
580     lw   $v0,24($fp)
581     addu     $v0,$v1,$v0
582     lb   $v0,0($v0)
583     addu     $v0,$a0,$v0
584     addu     $v0,$v0,-48
585     sw   $v0,28($fp)
586     lw   $v0,24($fp)
587     addu     $v0,$v0,1
588     sw   $v0,24($fp)
589     b    $L44
590 $L45:
591     lw   $v0,28($fp)
592     beq  $v0,$zero,$L50
593     lw   $a0,56($fp)
594     la   $t9,strlen
595     jal  $ra,$t9
596     move     $v1,$v0
597     lw   $v0,24($fp)
598     beq  $v0,$v1,$L50
599     b    $L49
```

```
600 $L50:
601     li   $v0,-1            # 0xffffffffffffffff
602     sw   $v0,36($fp)
603     b    $L43
604 $L49:
605     lw   $v0,24($fp)
606     addu     $v0,$v0,1
607     sw   $v0,24($fp)
608 $L51:
609     lw   $v1,56($fp)
610     lw   $v0,24($fp)
611     addu     $v0,$v1,$v0
612     lb   $v0,0($v0)
613     bne  $v0,$zero,$L53
614     b    $L52
615 $L53:
616     lw   $v1,56($fp)
617     lw   $v0,24($fp)
618     addu     $v0,$v1,$v0
619     lb   $v1,0($v0)
620     lw   $v0,_ctype_
621     addu     $v0,$v1,$v0
622     addu     $v0,$v0,1
623     lbu  $v0,0($v0)
624     srl  $v0,$v0,2
625     andi     $v0,$v0,0x1
626     bne  $v0,$zero,$L54
627     li   $v0,-1            # 0xffffffffffffffff
628     sw   $v0,36($fp)
629     b    $L43
630 $L54:
631     lw   $v1,32($fp)
632     move     $v0,$v1
633     sll  $v0,$v0,2
634     addu     $v0,$v0,$v1
635     sll  $a0,$v0,1
636     lw   $v1,56($fp)
637     lw   $v0,24($fp)
638     addu     $v0,$v1,$v0
639     lb   $v0,0($v0)
640     addu     $v0,$a0,$v0
641     addu     $v0,$v0,-48
642     sw   $v0,32($fp)
643     lw   $v0,24($fp)
644     addu     $v0,$v0,1
645     sw   $v0,24($fp)
646     b    $L51
647 $L52:
648     lw   $v0,32($fp)
649     bne  $v0,$zero,$L55
650     li   $v0,-1            # 0xffffffffffffffff
651     sw   $v0,36($fp)
652     b    $L43
653 $L55:
654     lw   $v1,60($fp)
655     lw   $v0,28($fp)
656     sw   $v0,0($v1)
```

```
657     lw   $v1,64($fp)
658     lw   $v0,32($fp)
659     sw   $v0,0($v1)
660     sw   $zero,36($fp)
661 $L43:
662     lw   $v0,36($fp)
663     move     $sp,$fp
664     lw   $ra,48($sp)
665     lw   $fp,44($sp)
666     addu     $sp,$sp,56
667     j    $ra
668     .end     set_resolution
669     .size    set_resolution, .-set_resolution
670     .rdata
671     .align   2
672 $LC18:
673     .ascii   "A\n\000"
674     .align   2
675 $LC19:
676     .ascii   "out of range\000"
677     .align   2
678 $LC20:
679     .ascii   "Re value not a number: <%s>\n\000"
680     .align   2
681 $LC21:
682     .ascii   "Complex number must follow the a+bi notation.\n\000"
683     .text
684     .align   2
685     .ent     set_complex
686 set_complex:
687     .frame   $fp,88,$ra      # vars= 48, regs= 3/0, args= 16, extra=
            8
688     .mask    0xd0000000,-8
689     .fmask   0x00000000,0
690     .set     noreorder
691     .cpload  $t9
692     .set     reorder
693     subu     $sp,$sp,88
694     .cprestore 16
695     sw   $ra,80($sp)
696     sw   $fp,76($sp)
697     sw   $gp,72($sp)
698     move     $fp,$sp
699     sw   $a0,88($fp)
700     sw   $a1,92($fp)
701     lw   $v0,88($fp)
702     lb   $v0,0($v0)
703     bne  $v0,$zero,$L57
704     la   $a0,$LC18
705     la   $t9,printf
706     jal  $ra,$t9
707     li   $v0,-1          # 0xffffffffffffffff
708     sw   $v0,64($fp)
709     b    $L56
710 $L57:
711     sw   $zero,24($fp)
712     sw   $zero,32($fp)
```

```
713     sw   $zero,36($fp)
714     sw   $zero,40($fp)
715     sw   $zero,44($fp)
716     li   $v0,1               # 0x1
717     sw   $v0,24($fp)
718 $L58:
719     lw   $v1,88($fp)
720     lw   $v0,24($fp)
721     addu     $v0,$v1,$v0
722     lb   $v1,0($v0)
723     li   $v0,43              # 0x2b
724     beq  $v1,$v0,$L59
725     lw   $v1,88($fp)
726     lw   $v0,24($fp)
727     addu     $v0,$v1,$v0
728     lb   $v1,0($v0)
729     li   $v0,45              # 0x2d
730     beq  $v1,$v0,$L59
731     lw   $v1,88($fp)
732     lw   $v0,24($fp)
733     addu     $v0,$v1,$v0
734     lb   $v0,0($v0)
735     bne  $v0,$zero,$L60
736     b    $L59
737 $L60:
738     lw   $v0,24($fp)
739     addu     $v0,$v0,1
740     sw   $v0,24($fp)
741     b    $L58
742 $L59:
743     lw   $v0,24($fp)
744     addu     $v0,$v0,1
745     move     $a0,$v0
746     la   $t9,malloc
747     jal  $ra,$t9
748     sw   $v0,48($fp)
749     sw   $zero,52($fp)
750 $L63:
751     lw   $v0,52($fp)
752     lw   $v1,24($fp)
753     slt  $v0,$v0,$v1
754     bne  $v0,$zero,$L66
755     b    $L64
756 $L66:
757     lw   $v1,48($fp)
758     lw   $v0,52($fp)
759     addu     $a0,$v1,$v0
760     lw   $v1,88($fp)
761     lw   $v0,52($fp)
762     addu     $v0,$v1,$v0
763     lbu  $v0,0($v0)
764     sb   $v0,0($a0)
765     lw   $v0,52($fp)
766     addu     $v0,$v0,1
767     sw   $v0,52($fp)
768     b    $L63
769 $L64:
```

```
770     lw   $v1,48($fp)
771     lw   $v0,52($fp)
772     addu     $v0,$v1,$v0
773     sb   $zero,0($v0)
774     la   $t9,__errno
775     jal  $ra,$t9
776     sw   $zero,0($v0)
777     addu     $v0,$fp,56
778     lw   $a0,48($fp)
779     move     $a1,$v0
780     la   $t9,strtod
781     jal  $ra,$t9
782     s.d  $f0,32($fp)
783     la   $t9,__errno
784     jal  $ra,$t9
785     lw   $v1,0($v0)
786     li   $v0,34              # 0x22
787     bne  $v1,$v0,$L67
788     la   $a0,$LC19
789     la   $t9,perror
790     jal  $ra,$t9
791     li   $v0,-1              # 0xffffffffffffffff
792     sw   $v0,64($fp)
793     b    $L56
794 $L67:
795     lw   $v0,56($fp)
796     lb   $v0,0($v0)
797     beq  $v0,$zero,$L68
798     la   $a0,$LC20
799     lw   $a1,56($fp)
800     la   $t9,printf
801     jal  $ra,$t9
802     lw   $a0,48($fp)
803     la   $t9,free
804     jal  $ra,$t9
805     li   $v0,-1              # 0xffffffffffffffff
806     sw   $v0,64($fp)
807     b    $L56
808 $L68:
809     lw   $a0,48($fp)
810     la   $t9,free
811     jal  $ra,$t9
812     lw   $v1,88($fp)
813     lw   $v0,24($fp)
814     addu     $v0,$v1,$v0
815     sw   $v0,60($fp)
816     lw   $v0,60($fp)
817     lb   $v0,0($v0)
818     beq  $v0,$zero,$L69
819     lw   $a0,60($fp)
820     la   $t9,strlen
821     jal  $ra,$t9
822     move     $v1,$v0
823     lw   $v0,60($fp)
824     addu     $v0,$v1,$v0
825     addu     $v0,$v0,-1
826     lb   $v1,0($v0)
```

```
827     li   $v0,105          # 0x69
828     beq  $v1,$v0,$L69
829     la   $a0,$LC21
830     la   $t9,printf
831     jal  $ra,$t9
832     li   $v0,-1           # 0xffffffffffffffff
833     sw   $v0,64($fp)
834     b    $L56
835 $L69:
836     addu     $v0,$fp,56
837     lw   $a0,60($fp)
838     move     $a1,$v0
839     la   $t9,strtod
840     jal  $ra,$t9
841     s.d  $f0,40($fp)
842     lw   $v0,56($fp)
843     lb   $v1,0($v0)
844     li   $v0,105          # 0x69
845     beq  $v1,$v0,$L70
846     li   $v0,-1           # 0xffffffffffffffff
847     sw   $v0,64($fp)
848     b    $L56
849 $L70:
850     lw   $v0,92($fp)
851     l.d  $f0,32($fp)
852     s.d  $f0,0($v0)
853     lw   $v0,92($fp)
854     l.d  $f0,40($fp)
855     s.d  $f0,8($v0)
856     sw   $zero,64($fp)
857 $L56:
858     lw   $v0,64($fp)
859     move     $sp,$fp
860     lw   $ra,80($sp)
861     lw   $fp,76($sp)
862     addu     $sp,$sp,88
863     j    $ra
864     .end     set_complex
865     .size    set_complex, .-set_complex
866     .rdata
867     .align   2
868 $LC22:
869     .ascii   "set_size val argument not a number:<%s>\n\000"
870     .text
871     .align   2
872     .ent     set_size
873 set_size:
874     .frame   $fp,48,$ra       # vars= 8, regs= 3/0, args= 16, extra=
            8
875     .mask    0xd0000000,-8
876     .fmask   0x00000000,0
877     .set     noreorder
878     .cpload  $t9
879     .set     reorder
880     subu     $sp,$sp,48
881     .cprestore 16
882     sw   $ra,40($sp)
```

```
883     sw   $fp,36($sp)
884     sw   $gp,32($sp)
885     move     $fp,$sp
886     sw   $a0,48($fp)
887     sw   $a1,52($fp)
888     la   $t9,__errno
889     jal  $ra,$t9
890     sw   $zero,0($v0)
891     lw   $a0,48($fp)
892     addu     $a1,$fp,24
893     la   $t9,strtod
894     jal  $ra,$t9
895     lw   $v0,52($fp)
896     s.d  $f0,0($v0)
897     la   $t9,__errno
898     jal  $ra,$t9
899     lw   $v1,0($v0)
900     li   $v0,34              # 0x22
901     bne  $v1,$v0,$L72
902     la   $a0,$LC19
903     la   $t9,perror
904     jal  $ra,$t9
905     li   $v0,-1              # 0xffffffffffffffff
906     sw   $v0,28($fp)
907     b    $L71
908 $L72:
909     lw   $v0,24($fp)
910     lb   $v0,0($v0)
911     beq  $v0,$zero,$L73
912     la   $a0,$LC22
913     lw   $a1,24($fp)
914     la   $t9,printf
915     jal  $ra,$t9
916     li   $v0,-1              # 0xffffffffffffffff
917     sw   $v0,28($fp)
918     b    $L71
919 $L73:
920     sw   $zero,28($fp)
921 $L71:
922     lw   $v0,28($fp)
923     move     $sp,$fp
924     lw   $ra,40($sp)
925     lw   $fp,36($sp)
926     addu     $sp,$sp,48
927     j    $ra
928     .end     set_size
929     .size    set_size, .-set_size
930     .rdata
931     .align   2
932 $LC24:
933     .ascii   "P2\n\000"
934     .align   2
935 $LC25:
936     .ascii   "%d\n\000"
937     .align   2
938 $LC26:
939     .ascii   "%d \000"
```

```
940     .align  2
941 $LC27:
942     .ascii  "\n\000"
943     .align  3
944 $LC23:
945     .word   0
946     .word   1073741824
947     .text
948     .align  2
949     .ent    drawJulia
950 drawJulia:
951     .frame  $fp,136,$ra     # vars= 64, regs= 3/0, args= 48, extra=
            8
952     .mask   0xd0000000,-8
953     .fmask  0x00000000,0
954     .set    noreorder
955     .cpload $t9
956     .set    reorder
957     subu    $sp,$sp,136
958     .cprestore 48
959     sw  $ra,128($sp)
960     sw  $fp,124($sp)
961     sw  $gp,120($sp)
962     move    $fp,$sp
963     sw  $a0,136($fp)
964     sw  $a1,140($fp)
965     sw  $a2,144($fp)
966     sw  $a3,148($fp)
967     l.s $f0,136($fp)
968     cvt.d.w $f2,$f0
969     l.d $f0,144($fp)
970     div.d   $f0,$f0,$f2
971     s.d $f0,56($fp)
972     l.s $f0,140($fp)
973     cvt.d.w $f2,$f0
974     l.d $f0,152($fp)
975     div.d   $f0,$f0,$f2
976     s.d $f0,64($fp)
977     l.d $f2,144($fp)
978     l.d $f0,$LC23
979     div.d   $f2,$f2,$f0
980     l.d $f0,160($fp)
981     sub.d   $f0,$f0,$f2
982     s.d $f0,72($fp)
983     l.d $f2,152($fp)
984     l.d $f0,$LC23
985     div.d   $f2,$f2,$f0
986     l.d $f0,168($fp)
987     add.d   $f0,$f0,$f2
988     s.d $f0,80($fp)
989     lw  $a0,196($fp)
990     la  $a1,$LC24
991     la  $t9,fprintf
992     jal $ra,$t9
993     lw  $a0,196($fp)
994     la  $a1,$LC25
995     lw  $a2,136($fp)
```

```
996     la   $t9,fprintf
997     jal  $ra,$t9
998     lw   $a0,196($fp)
999     la   $a1,$LC25
1000    lw   $a2,140($fp)
1001    la   $t9,fprintf
1002    jal  $ra,$t9
1003    lw   $a0,196($fp)
1004    la   $a1,$LC25
1005    lw   $a2,192($fp)
1006    la   $t9,fprintf
1007    jal  $ra,$t9
1008    sw   $zero,88($fp)
1009 $L75:
1010    lw   $v0,88($fp)
1011    lw   $v1,140($fp)
1012    slt  $v0,$v0,$v1
1013    bne  $v0,$zero,$L78
1014    b    $L74
1015 $L78:
1016    sw   $zero,92($fp)
1017 $L79:
1018    lw   $v0,92($fp)
1019    lw   $v1,136($fp)
1020    slt  $v0,$v0,$v1
1021    bne  $v0,$zero,$L82
1022    b    $L80
1023 $L82:
1024    addu     $v0,$fp,96
1025    l.d  $f0,56($fp)
1026    s.d  $f0,16($sp)
1027    l.d  $f0,64($fp)
1028    s.d  $f0,24($sp)
1029    l.d  $f0,72($fp)
1030    s.d  $f0,32($sp)
1031    l.d  $f0,80($fp)
1032    s.d  $f0,40($sp)
1033    move     $a0,$v0
1034    lw   $a1,92($fp)
1035    lw   $a2,88($fp)
1036    la   $t9,complex_map
1037    jal  $ra,$t9
1038    lw   $v0,192($fp)
1039    sw   $v0,112($fp)
1040    sw   $zero,116($fp)
1041 $L83:
1042    lw   $v0,192($fp)
1043    addu     $v1,$v0,-1
1044    lw   $v0,116($fp)
1045    slt  $v0,$v0,$v1
1046    bne  $v0,$zero,$L86
1047    b    $L84
1048 $L86:
1049    addu     $v0,$fp,96
1050    move     $a0,$v0
1051    la   $t9,complex_abs
1052    jal  $ra,$t9
```

```
1053      mov.d     $f2,$f0
1054      l.d  $f0,$LC23
1055      c.lt.d    $f0,$f2
1056      bc1t      $L88
1057      b    $L87
1058  $L88:
1059      lw   $v0,116($fp)
1060      sw   $v0,112($fp)
1061      b    $L84
1062  $L87:
1063      addu      $v1,$fp,96
1064      lw   $v0,104($fp)
1065      sw   $v0,16($sp)
1066      lw   $v0,108($fp)
1067      sw   $v0,20($sp)
1068      lw   $a2,96($fp)
1069      lw   $a3,100($fp)
1070      move      $a0,$v1
1071      la   $t9,complex_mult
1072      jal  $ra,$t9
1073      addu      $v1,$fp,96
1074      lw   $v0,184($fp)
1075      sw   $v0,16($sp)
1076      lw   $v0,188($fp)
1077      sw   $v0,20($sp)
1078      lw   $a2,176($fp)
1079      lw   $a3,180($fp)
1080      move      $a0,$v1
1081      la   $t9,complex_add
1082      jal  $ra,$t9
1083      lw   $v0,116($fp)
1084      addu      $v0,$v0,1
1085      sw   $v0,116($fp)
1086      b    $L83
1087  $L84:
1088      lw   $a0,196($fp)
1089      la   $a1,$LC26
1090      lw   $a2,112($fp)
1091      la   $t9,fprintf
1092      jal  $ra,$t9
1093      lw   $v0,92($fp)
1094      addu      $v0,$v0,1
1095      sw   $v0,92($fp)
1096      b    $L79
1097  $L80:
1098      lw   $a0,196($fp)
1099      la   $a1,$LC27
1100      la   $t9,fprintf
1101      jal  $ra,$t9
1102      lw   $v0,88($fp)
1103      addu      $v0,$v0,1
1104      sw   $v0,88($fp)
1105      b    $L75
1106  $L74:
1107      move      $sp,$fp
1108      lw   $ra,128($sp)
1109      lw   $fp,124($sp)
```

```
1110        addu    $sp,$sp,136
1111        j       $ra
1112        .end    drawJulia
1113        .size   drawJulia, .-drawJulia
1114        .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

../MIPS32/app.s

B. Enunciado original

Univesidad de Buenos Aires - FIUBA
66:20 Organización de Computadoras
Trabajo práctico 0: Infraestructura básica
2^o cuatrimestre de 2016

`$Date: 2016/08/30 04:13:03 $`

1. Objetivos

Familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementando un programa y su correspondiente documentación que resuelvan el problema descripto más abajo.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes, un informe impreso de acuerdo con lo que mencionaremos en la sección 6, y con una copia digital de los archivos fuente necesarios para compilar el trabajo.

4. Recursos

Usaremos el programa GXemul [1] para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión reciente del sistema operativo NetBSD [2].

Durante la primera clase del curso presentaremos brevemente los pasos necesarios para la instalación y configuración del entorno de desarrollo.

5. Programa

Se trata de diseñar un programa que permita dibujar el conjunto de Julia [3] y sus vecindades, en lenguaje C, correspondiente a un polinomio cuadrático.

El mismo recibirá, por línea de comando, una serie de parámetros describiendo la región del plano complejo, las características del archivo imagen a generar, y el parámetro c.

No deberá interactuar con el usuario, ya que no se trata de un programa interactivo, sino más bien de una herramienta de procesamiento *batch*. Al finalizar la ejecución, y volver al sistema operativo, el programa habrá dibujado el fractal en el archivo de salida.

El formato gráfico a usar es PGM o *portable gray map* [4], un formato simple para describir imágenes digitales monocromáticas.

5.1. Algoritmo

El algoritmo básico es simple: para algunos puntos z de la región del plano que estamos procesando haremos un cálculo repetitivo. Terminado el cálculo, asignamos el nivel de intensidad del pixel en base a la condición de corte de ese cálculo.

El color de cada punto representa la "velocidad de escape" asociada con ese número complejo: blanco para aquellos puntos que pertenecen al conjunto (y por ende la "cuenta" permanece acotada), y tonos gradualmente más oscuros para los puntos divergentes, que no pertenezcan al conjunto.

Más específicamente: para cada pixel de la pantalla, tomaremos su punto medio, expresado en coordenadas complejas, $z = z_{re} + z_{im}i$. A continuación, iteramos sobre $z_{n+1} = z_n^2 + c$, con $z_0 = z$. Cortamos la iteración cuando $|z_n| > 2$, o después de `N` iteraciones.

En pseudo código:

```
para cada pixel $p {
        $z = complejo asociado a $p;
        for ($i = 0; $i < $N - 1; ++$i) {
                if (abs($z) > 2)
                        break;
                $z = $z * $z + $c;
        }
        dibujar el punto p con brillo $i;
}
```

Notar que c es un parámetro del programa.

Así tendremos, al finalizar, una representación visual de la cantidad de ciclos de cómputo realizados hasta alcanzar la condición de escape (ver figura 1).

5.2. Interfaz

A fin de facilitar el intercambio de código *ad-hoc*, normalizaremos algunas de las opciones que deberán ser provistas por el programa:

- `-r`: permite cambiar la resolución de la imagen generada. El valor por defecto será de 640x480 puntos.
- `-c`: para especificar el centro de la imagen, el punto central de la porción del plano complejo dibujada, expresado en forma binómica (i.e. $a + bi$). Por defecto usaremos $0 + 0i$.
- `-C`: determina el parámetro c, también expresado en forma binómica. El valor por defecto será $0{,}285 - 0{,}01i$.
- `-w`: especifica el ancho del rectángulo que contiene la región del plano complejo que estamos por dibujar. Valor por defecto: 4.

- `-H`: sirve, en forma similar, para especificar el alto del rectángulo a dibujar. Valor por defecto: 4.
- `-o`: permite colocar la imagen de salida, (en formato PGM [4]) en el archivo pasado como argumento; o por salida estándar `-stdout-` si el argumento es "-".

5.3. Casos de prueba

Es necesario que el informe trabajo práctico incluya una sección dedicada a verificar el funcionamiento del código implementado.

En el caso del TP 0, será necesario escribir pruebas orientadas a probar el programa completo, ejercitando los casos más comunes de funcionamiento, los casos de borde, y también casos de error.

Incluimos en el apéndice A algunos ejemplos de casos de interés, orientados a ejercitar algunos errores y condiciones de borde.

5.4. Ejemplos

Generamos un dibujo usando los valores por defecto, barriendo la región rectangular del plano comprendida entre los vértices $-2+2i$ y $+2-2i$.

```
$ tp0 -o uno.pgm
```

La figura 1 muestra la imagen `uno.pgm`.

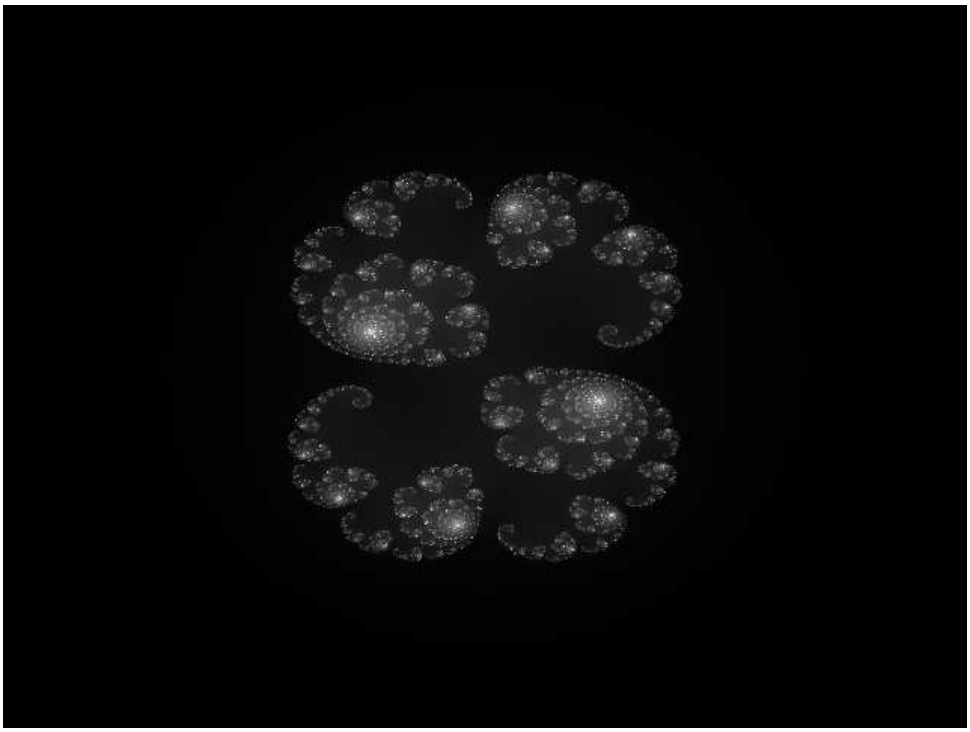

Figura 1: Región barrida por defecto.

A continuación, hacemos *zoom* sobre la región centrada en $+0{,}282 - 0{,}01i$, usando un rectángulo de 0,005 unidades de lado. El resultado podemos observarlo en la figura 2.

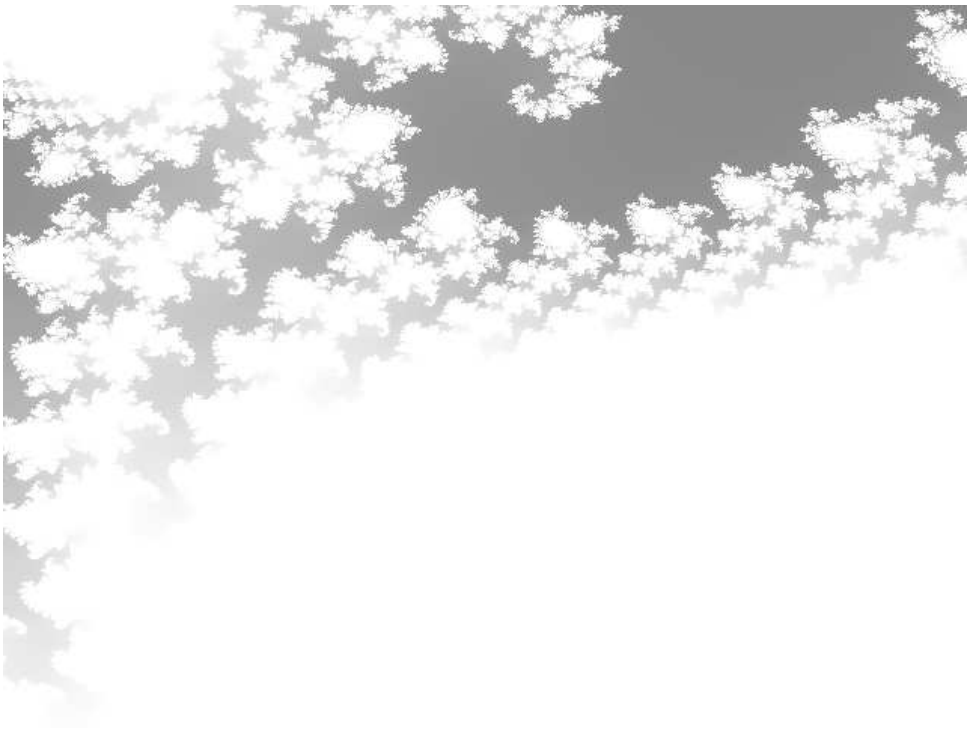

Figura 2: Región comprendida entre $0{,}2795 - 0{,}0075i$ y $0{,}2845 - 0{,}0125i$.

```
$ tp0 -c +0.282-0.01i -w 0.005 -H 0.005 -o dos.pgm
```

6. Informe

El informe deberá incluir:

- Documentación relevante al diseño e implementación del programa.
- Documentación relevante al proceso de compilación: cómo obtener el ejecutable a partir de los archivos fuente.
- Las corridas de prueba, con los comentarios pertinentes.
- El código fuente, en lenguaje C.
- Este enunciado.

7. Fechas

Fecha de vencimiento: Martes 26/9.

Referencias

[1] GXemul, `http://gavare.se/gxemul/`.

[2] The NetBSD project.
http://www.netbsd.org/.

[3] http://en.wikipedia.org/wiki/Julia_set (Wikipedia).

[4] PGM format specification.
http://netpbm.sourceforge.net/doc/pgm.html.

A. Algunos casos de prueba

1. Generamos una imagen de 1 punto de lado, centrada en el orígen del plano complejo:

```
$ tp0 -c 0.01+0i -r 1x1 -o -
P2
1
1
255
255
```

   Notar que el resultado es correcto, ya que este punto pertenece al conjunto de Julia.

2. Repetimos el experimento, pero nos centramos ahora en un punto que *seguro* no pertenece al conjunto:

```
$ tp0 -c 10+0i -r 1x1 -o -
P2
1
1
255
0
```

   Notar que el resultado es correcto, ya que este punto no pertenece al conjunto de Julia.

3. Imagen imposible:

```
$ tp0 -c 0+0i -r 0x1 -o -
Usage:
  tp0 -h
  tp0 -V
...
```

4. Archivo de salida imposible:

```
$ tp0 -o /tmp
fatal: cannot open output file.
```

5. Coordenadas complejas imposibles:

```
$ tp0 -c 1+3 -o -
fatal: invalid center specification.
```

6. Argumentos de línea de comando vacíos,

```
$ tp0 -c "" -o -
fatal: invalid center specification.
```