U.B.A. FACULTAD DE INGENIERÍA

Departamento de Electrónica
Organización de Computadoras 66-20
Ing. Informática

Trabajo práctico Nº01
Assembly Mips

Apellido y Nombre:	*Padrón:*
Cabrera, Jorge	93310
Capolupo, Mauro	90283

Fecha de Entrega : 20/09/2016
Fecha de Reentrega :
Fecha de Aprobación :
Calificación :
Firma de Aprobación :

1 Diseño e implementación del programa

El programa implementa un algoritmo sencillo que permite dibujar el conjunto de Julia y sus vecindades correspondiente a un polinomio cuadrático, según un determinado conjunto de parámetros que serán ingresados por línea de comando.

Su implementación puede dividirse en dos etapas:

- Etapa 1: se lee los parámetros recibidos por parámetros y se los parsea.
- Etapa 2: se procede a ejecutar el algoritmo que calcula los puntos que, dado un C, pertenecen al conjunto de julia.

1.1 Parámetros

- -r: setea la resolución de la imágen generada
- -c: especifica el centro de la imágen
- -C: indica el complejo que se le va a sumar en cada iteración al número complejo asociado a cada pixel,

$$z^2 + C$$

- -w: ancho del ractángulo de la región del plano complejo a dibujar
- -H: alto del ractángulo de la región del plano complejo a dibujar
- -o: permite colocar la imagen de salida en el archivo pasado como parámetro o stdout en caso de que el argumento sea '-'

2 Comando(s) para compilar el programa en NetBSD

gcc -Wall -g -o orga2016TP0 orga2016TP0.c

3 Comando(s) para ejecutar el programa en NetBSD

./orga2016TP0 -o Julia.pgm

4 Pruebas

Prueba 1

Generamos el conjunto Julia con parámetros válidos

./orga2016TP0 -r 640x480 -c 1-1i -w 4 -H 2 -o Julia.pgm

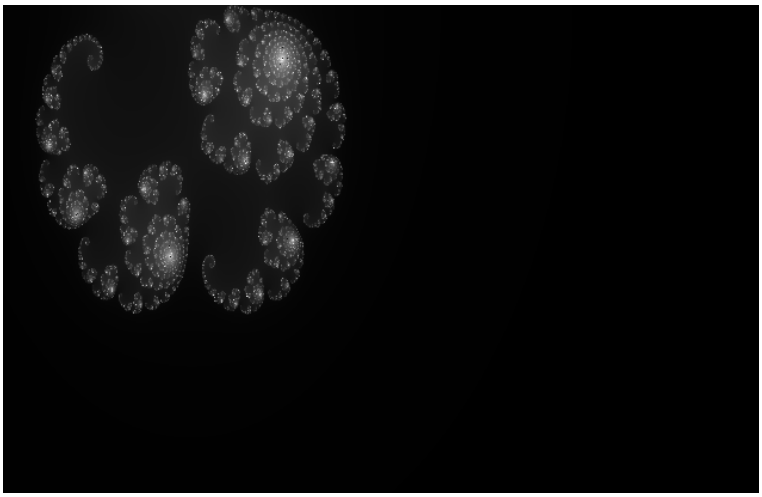

Figure 1: Prueba 1

Prueba 2

Generamos el conjunto Julia con parámetros válidos

./orga2016TP0 -C 0.376-0.1566i -c 0+0i -w 4 -H 4 -o Julia2.pgm

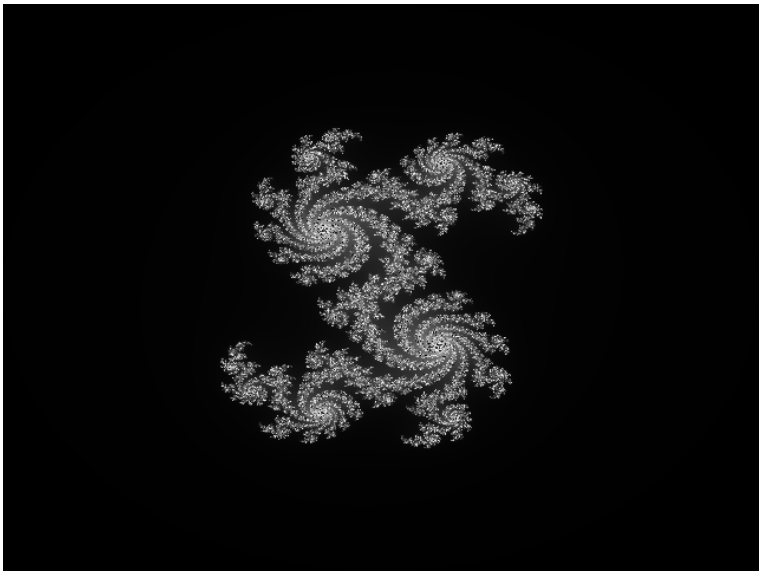

Figure 2: Prueba 2

Prueba 3

Generamos el conjunto Julia con parámetros válidos

./orga2016TP0 -r 480x240 -c -0.1+0.1i -w 4 -H 2 -o Julia.pgm

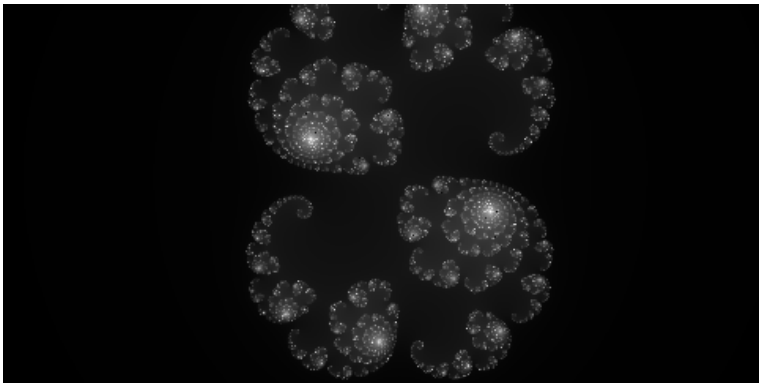

Figure 3: Prueba 3

Prueba 4

Generamos el conjunto Julia con parámetros válidos sin especificar c

./orga2016TP0 -r 640x480 -w 4 -H 2 -o Julia.pgm

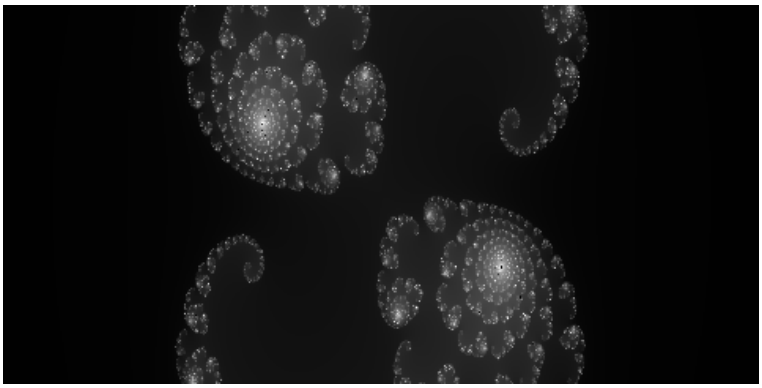

Figure 4: Prueba 4

5 Pruebas casos de errores

Prueba 1

Parámetro r inválido

./orga2016TP0 -r 640-480 -c 1-1i -w 4 -H 2 -o Julia.pgm

Se esperaba que el caracter delimitador sea 'x'. El mensaje que veremos por pantalla será:

Fatal: invalid r specification.

Prueba 2

Parámetro w inválido

./orga2016TP0 -r 640x480 -c 1-1i -w A -H 2 -o Julia.pgm

Los valores tanto de 'w' como de 'H' deben ser valores enteros.

Fatal: invalid w specification.

Prueba 3

Parámetro c inválido

./orga2016TP0 -r 640x480 -c a-bi -w 2 -H 2 -o Julia.pgm

Los valores correspondientes a la parte real e imaginaria del parámetro complejo c deben ser reales.

Fatal: invalid c specification.

Prueba 4

Parámetro r inválido

./orga2016TP0 -r 0x480 -c 2-3i -w 2 -H 2 -o Julia.pgm

Tanto el ancho como el alto de la imágen deben ser mayores a 0

Fatal: invalid r specification.

Prueba 5

Parámetro c inválido

./orga2016TP0 -r 240x480 -c 2-3 -w 2 -H 2 -o Julia.pgm

El valor del parámetro c debe ser un número complejo

Fatal: invalid c specification.

6 Codigo

6.1 Código C

```
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <string.h>

char getDelimSymbol(char* cvalue);
float getReValue(char* string, char delim);
float getImValue(char* string, char delim);
bool isComplexValue(char* string);
int analyzerComplexParameter(char* cValue, float* cRe, float* cIm);
int setResolution(char* rvalue, float* width, float* height);
void drawJuliaSet(FILE* pgmFile, float resW, float resH, float recW, float recH,
                float compRe, float compIm, float centerRe, float centerIm);

int main(int argc, char *argv[]) {
        if (argc == 1) {
                printf("No options given\n");
                //return;
        }
        //global variables
        int c;
        float resW = 640;
        float resH = 480;
        float centerRe = 0;
        float centerIm = 0;
        float complexRe = 0.285;
        float complexIm = 0.01;
        FILE * pgmFile;
        float recW = 4;
        float recH = 4;

        //args values
        char *ovalue = NULL;

        //read args params
        while ((c = getopt(argc, argv, "r:c:C:w:H:o:")) != -1) {
                switch (c) {
                case 'r':
                        //resolution value
                        if(setResolution(optarg, &resW, &resH) != 0){
                                printf("Fatal: invalid r specification.\n");
                                return -1;
                        }
                        break;
                case 'c':
                        //image center value
                        if (analyzerComplexParameter(optarg, &centerRe, &centerIm) != 0) {
                                printf("Fatal: invalid c specification.\n");
                                return -1;
                        }
                        break;
                case 'C':
                        //C parameter value
                        if (analyzerComplexParameter(optarg, &complexRe, &complexIm) != 0) {
                                printf("Fatal: invalid C specification.\n");
                                return -1;
                        }
                        break;
                case 'w':
                        //width value
                        if(!isdigit(*optarg)){
                                printf("Fatal: invalid w specification.\n");
                                return -1;
                        }
                        recW = atof(optarg);
                        break;
                case 'H':
                        //high value
                        if(!isdigit(*optarg)){
                                printf("Fatal: invalid H specification.\n");
                                return -1;
                        }
                        recH = atof(optarg);
                        break;
                case 'o':
                        ovalue = optarg;
                        //file name value
                        if (strcmp(ovalue, "-") == 0) {
                                pgmFile = stdout;
                        } else {
```

```
                                        pgmFile = fopen(ovalue, "wb");
                                        if (pgmFile == NULL) {
                                                printf("Fatal: cannot open output file.");
                                                return -1;
                                        }
                                }
                                break;
                        }
                }

                if (!ovalue) {
                        printf("Fatal: -o parameter is mandatory.");
                        return -1;
                }

                drawJuliaSet(pgmFile, resW, resH, recW, recH, complexRe, complexIm,
                                centerRe, centerIm);

                return 0;
}

void drawJuliaSet(FILE* pgmFile, float resW, float resH, float recW, float recH,
                float compRe, float compIm, float centerRe, float centerIm) {

                // a + bi
                float a, b;

                // Start at negative half the rectangleDrawWidth and resolutionHeight
                float xmin = -recW / 2 + centerRe;
                float ymin = -recH / 2 + centerIm;

                // x goes from xmin to xmax
                float xmax = xmin + recW;
                // y goes from ymin to ymax
                float ymax = ymin + recH;

                // Calculate amount we increment x,y for each pixel
                float dx = (xmax - xmin) / (resW);
                float dy = (ymax - ymin) / (resH);

                int maxIterations = 1000;

                char header[100];
                sprintf(header, "P2\n# Julia Set image\n %f %f \n255\n", resW, resH);
                fputs(header, pgmFile);

                //loop through every pixel
                // Start y
                float y = ymin;
                int j, i;
                for (j = 0; j < resH; j++) {
                        // Start x
                        float x = xmin;
                        for (i = 0; i < resW; i++) {
                                a = x;
                                b = y;
                                int n;
                                //start the iteration process
                                for (n = 0; i < maxIterations; n++) {
                                        float aa = a * a;
                                        float bb = b * b;

                                        if ((aa + bb) > 4)
                                                break;

                                        float twoab = 2.0 * a * b;
                                        a = aa - bb + compRe;
                                        b = twoab + compIm;
                                }
                                char str[15];
                                sprintf(str, "%d ", n);
                                fputs(str, pgmFile);
                                x += dx;
                        }
                        fputs("\n", pgmFile);
                        y += dy;
                }
}

char getDelimSymbol(char* cvalue) {
                int len = strlen(cvalue);
                char delim;
                while (len > 0) {
                        char value = cvalue[len - 1];
                        if (!isdigit(value) && value != '.' && !isalpha(value)) {
                                delim = value;
                                break;
                        }
                        len = len - 1;
                }
                return delim;
```

```
}

float getReValue(char* string, char delim) {
        //10 will be the maximum number of digits
        char token[10] = "";
        int len = 0;
        while (len < strlen(string)) {
                char value = string[len];
                if (len > 0 && value == delim) {
                        break;
                }else if(isdigit(value) || value == '.'){
                        *(&token[len]) = value;
                }else{
                        return -1;
                }
                len++;
        }
        return atof(token);
}

float getImValue(char* string, char delim) {
        //10 will be the maximum number of digits
        int len = strlen(string);
        char token[10] = "";
        while (len > 0) {
                char value = string[len - 1];
                if (value == delim) {
                        break;
                }else if(isdigit(value) || value == '.'){
                        token[len - 1] = value;
                }else{
                        return -1;
                }
                len--;
        }
        return atof(&token[len]);
}

bool isComplexValue(char* string) {
        int len = strlen(string);
        const char * last = &string[len - 1];
        if (strcmp(last, "i") != 0) {
                return false;
        }
        return true;
}

int analyzerComplexParameter(char* cValue, float* cRe, float* cIm) {
        if (strcmp(cValue, "") == 0) {
                return -1;
        }
        if (!isComplexValue(cValue)) {
                return -1;
        }
        //extract trailing i
        strtok(cValue, "i");
        char delim = getDelimSymbol(cValue);
        *cRe = getReValue(cValue, delim);
        *cIm = getImValue(cValue, delim);
        if(*cRe == -1 || *cIm == -1){
                return -1;
        }
        return 0;
}

int setResolution(char* rvalue, float* width, float* height) {
        char *delim = "x";
        char *token = strtok(rvalue, delim);
        *width = atof(token);
        token = strtok(0, delim);
        if(!token){
                return -1;
        }
        *height = atof(token);
        if(!(*width > 0) || !(*height>0)){
                return -1;
        }
        return 0;
}
```

6.2 Código MIPS

```
        .file   1 "orga2016TP0.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .rdata
        .align  2
$LC0:
        .ascii  "No options given\n\000"
        .align  2
$LC6:
        .ascii  "r:c:C:w:H:o:\000"
        .align  2
$LC7:
        .ascii  "Fatal: invalid r specification.\n\000"
        .align  2
$LC8:
        .ascii  "Fatal: invalid c specification.\n\000"
        .align  2
$LC9:
        .ascii  "Fatal: invalid C specification.\n\000"
        .align  2
$LC10:
        .ascii  "Fatal: invalid w specification.\n\000"
        .align  2
$LC11:
        .ascii  "Fatal: invalid H specification.\n\000"
        .align  2
$LC12:
        .ascii  "-\000"
        .align  2
$LC13:
        .ascii  "wb\000"
        .align  2
$LC14:
        .ascii  "Fatal: cannot open output file.\000"
        .align  2
$LC15:
        .ascii  "Fatal: -o parameter is mandatory.\000"
        .align  2
$LC1:
        .word   1142947840
        .align  2
$LC2:
        .word   1139802112
        .align  2
$LC3:
        .word   1049750405
        .align  2
$LC4:
        .word   1008981770
        .align  2
$LC5:
        .word   1082130432
        .text
        .align  2
        .globl  main
        .ent    main
main:
        .frame  $fp,120,$31             # vars= 56, regs= 3/0, args= 40, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $25
        .set    reorder
        subu    $sp,$sp,120
        .cprestore 40
        sw      $31,112($sp)
        sw      $fp,108($sp)
        sw      $28,104($sp)
        move    $fp,$sp
        sw      $4,120($fp)
        sw      $5,124($fp)
        lw      $3,120($fp)
        li      $2,1                    # 0x1
        bne     $3,$2,$L18
        la      $4,$LC0
        la      $25,printf
        jal     $31,$25
$L18:
        l.s     $f0,$LC1
        s.s     $f0,52($fp)
        l.s     $f0,$LC2
        s.s     $f0,56($fp)
        sw      $0,60($fp)
        sw      $0,64($fp)
        l.s     $f0,$LC3
        s.s     $f0,68($fp)
        l.s     $f0,$LC4
```

```
        s.s     $f0,72($fp)
        l.s     $f0,$LC5
        s.s     $f0,80($fp)
        l.s     $f0,$LC5
        s.s     $f0,84($fp)
        sw      $0,88($fp)
$L19:
        lw      $4,120($fp)
        lw      $5,124($fp)
        la      $6,$LC6
        la      $25,getopt
        jal     $31,$25
        sw      $2,48($fp)
        lw      $3,48($fp)
        li      $2,-1                   # 0xffffffffffffffff
        bne     $3,$2,$L21
        b       $L20
$L21:
        lw      $2,48($fp)
        addu    $2,$2,-67
        sw      $2,96($fp)
        lw      $3,96($fp)
        sltu    $2,$3,53
        beq     $2,$0,$L19
        lw      $2,96($fp)
        sll     $3,$2,2
        la      $2,$L37
        addu    $2,$3,$2
        lw      $2,0($2)
        .cpadd  $2
        j       $2
        .rdata
        .align  2
$L37:
        .gpword $L27
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L31
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L25
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L33
        .gpword $L19
        .gpword $L19
        .gpword $L23
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L19
        .gpword $L29
        .text
$L23:
        addu    $2,$fp,52
        addu    $3,$fp,56
```

```
        lw      $4,optarg
        move    $5,$2
        move    $6,$3
        la      $25,setResolution
        jal     $31,$25
        beq     $2,$0,$L19
        la      $4,$LC7
        la      $25,printf
        jal     $31,$25
        li      $2,-1                   # 0xffffffffffffffff
        sw      $2,92($fp)
        b       $L17
$L25:
        addu    $2,$fp,60
        addu    $3,$fp,64
        lw      $4,optarg
        move    $5,$2
        move    $6,$3
        la      $25,analyzerComplexParameter
        jal     $31,$25
        beq     $2,$0,$L19
        la      $4,$LC8
        la      $25,printf
        jal     $31,$25
        li      $3,-1                   # 0xffffffffffffffff
        sw      $3,92($fp)
        b       $L17
$L27:
        addu    $2,$fp,68
        addu    $3,$fp,72
        lw      $4,optarg
        move    $5,$2
        move    $6,$3
        la      $25,analyzerComplexParameter
        jal     $31,$25
        beq     $2,$0,$L19
        la      $4,$LC9
        la      $25,printf
        jal     $31,$25
        li      $2,-1                   # 0xffffffffffffffff
        sw      $2,92($fp)
        b       $L17
$L29:
        lw      $2,optarg
        lb      $3,0($2)
        lw      $2,_ctype_
        addu    $2,$3,$2
        addu    $2,$2,1
        lbu     $2,0($2)
        srl     $2,$2,2
        andi    $2,$2,0x1
        bne     $2,$0,$L30
        la      $4,$LC10
        la      $25,printf
        jal     $31,$25
        li      $3,-1                   # 0xffffffffffffffff
        sw      $3,92($fp)
        b       $L17
$L30:
        lw      $4,optarg
        la      $25,atof
        jal     $31,$25
        cvt.s.d $f0,$f0
        s.s     $f0,80($fp)
        b       $L19
$L31:
        lw      $2,optarg
        lb      $3,0($2)
        lw      $2,_ctype_
        addu    $2,$3,$2
        addu    $2,$2,1
        lbu     $2,0($2)
        srl     $2,$2,2
        andi    $2,$2,0x1
        bne     $2,$0,$L32
        la      $4,$LC11
        la      $25,printf
        jal     $31,$25
        li      $2,-1                   # 0xffffffffffffffff
        sw      $2,92($fp)
        b       $L17
$L32:
        lw      $4,optarg
        la      $25,atof
        jal     $31,$25
        cvt.s.d $f0,$f0
        s.s     $f0,84($fp)
        b       $L19
$L33:
        lw      $2,optarg
        sw      $2,88($fp)
```

```
        lw      $4,88($fp)
        la      $5,$LC12
        la      $25,strcmp
        jal     $31,$25
        bne     $2,$0,$L34
        la      $2,__sF+88
        sw      $2,76($fp)
        b       $L19
$L34:
        lw      $4,88($fp)
        la      $5,$LC13
        la      $25,fopen
        jal     $31,$25
        sw      $2,76($fp)
        lw      $2,76($fp)
        bne     $2,$0,$L19
        la      $4,$LC14
        la      $25,printf
        jal     $31,$25
        li      $3,-1                   # 0xffffffffffffffff
        sw      $3,92($fp)
        b       $L17
$L20:
        lw      $2,88($fp)
        bne     $2,$0,$L39
        la      $4,$LC15
        la      $25,printf
        jal     $31,$25
        li      $2,-1                   # 0xffffffffffffffff
        sw      $2,92($fp)
        b       $L17
$L39:
        l.s     $f0,84($fp)
        s.s     $f0,16($sp)
        l.s     $f0,68($fp)
        s.s     $f0,20($sp)
        l.s     $f0,72($fp)
        s.s     $f0,24($sp)
        l.s     $f0,60($fp)
        s.s     $f0,28($sp)
        l.s     $f0,64($fp)
        s.s     $f0,32($sp)
        lw      $4,76($fp)
        lw      $5,52($fp)
        lw      $6,56($fp)
        lw      $7,80($fp)
        la      $25,drawJuliaSet
        jal     $31,$25
        sw      $0,92($fp)
$L17:
        lw      $2,92($fp)
        move    $sp,$fp
        lw      $31,112($sp)
        lw      $fp,108($sp)
        addu    $sp,$sp,120
        j       $31
        .end    main
        .size   main, .-main
        .rdata
        .align  2
$LC17:
        .ascii  "P2\n"
        .ascii  "# Julia Set image\n"
        .ascii  " %f %f \n"
        .ascii  "255\n\000"
        .align  2
$LC19:
        .ascii  "%d \000"
        .align  2
$LC20:
        .ascii  "\n\000"
        .align  2
$LC16:
        .word   1073741824
        .align  2
$LC18:
        .word   1082130432
        .text
        .align  2
        .globl  drawJuliaSet
        .ent    drawJuliaSet
drawJuliaSet:
        .frame  $fp,240,$31             # vars= 192, regs= 3/0, args= 24, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $25
        .set    reorder
        subu    $sp,$sp,240
        .cprestore 24
        sw      $31,232($sp)
```

```
        sw      $fp,228($sp)
        sw      $28,224($sp)
        move    $fp,$sp
        sw      $4,240($fp)
        sw      $5,244($fp)
        sw      $6,248($fp)
        sw      $7,252($fp)
        l.s     $f0,252($fp)
        neg.s   $f2,$f0
        l.s     $f0,$LC16
        div.s   $f2,$f2,$f0
        l.s     $f0,268($fp)
        add.s   $f0,$f2,$f0
        s.s     $f0,40($fp)
        l.s     $f0,256($fp)
        neg.s   $f2,$f0
        l.s     $f0,$LC16
        div.s   $f2,$f2,$f0
        l.s     $f0,272($fp)
        add.s   $f0,$f2,$f0
        s.s     $f0,44($fp)
        l.s     $f2,40($fp)
        l.s     $f0,252($fp)
        add.s   $f0,$f2,$f0
        s.s     $f0,48($fp)
        l.s     $f2,44($fp)
        l.s     $f0,256($fp)
        add.s   $f0,$f2,$f0
        s.s     $f0,52($fp)
        l.s     $f2,48($fp)
        l.s     $f0,40($fp)
        sub.s   $f2,$f2,$f0
        l.s     $f0,244($fp)
        div.s   $f0,$f2,$f0
        s.s     $f0,56($fp)
        l.s     $f2,52($fp)
        l.s     $f0,44($fp)
        sub.s   $f2,$f2,$f0
        l.s     $f0,248($fp)
        div.s   $f0,$f2,$f0
        s.s     $f0,60($fp)
        li      $2,1000                 # 0x3e8
        sw      $2,64($fp)
        addu    $2,$fp,72
        l.s     $f0,244($fp)
        cvt.d.s $f2,$f0
        l.s     $f0,248($fp)
        cvt.d.s $f0,$f0
        s.d     $f0,16($sp)
        move    $4,$2
        la      $5,$LC17
        mfc1    $6,$f2
        mfc1    $7,$f3
        la      $25,sprintf
        jal     $31,$25
        addu    $2,$fp,72
        move    $4,$2
        lw      $5,240($fp)
        la      $25,fputs
        jal     $31,$25
        l.s     $f0,44($fp)
        s.s     $f0,176($fp)
        sw      $0,180($fp)
$L41:
        l.s     $f0,180($fp)
        cvt.s.w $f2,$f0
        l.s     $f0,248($fp)
        c.lt.s  $f2,$f0
        bc1t    $L44
        b       $L40
$L44:
        l.s     $f0,40($fp)
        s.s     $f0,188($fp)
        sw      $0,184($fp)
$L45:
        l.s     $f0,184($fp)
        cvt.s.w $f2,$f0
        l.s     $f0,244($fp)
        c.lt.s  $f2,$f0
        bc1t    $L48
        b       $L46
$L48:
        l.s     $f0,188($fp)
        s.s     $f0,32($fp)
        l.s     $f0,176($fp)
        s.s     $f0,36($fp)
        sw      $0,192($fp)
$L49:
        lw      $2,184($fp)
        lw      $3,64($fp)
        slt     $2,$2,$3
```

```
        bne     $2,$0,$L52
        b       $L50
$L52:
        l.s     $f2,32($fp)
        l.s     $f0,32($fp)
        mul.s   $f0,$f2,$f0
        s.s     $f0,196($fp)
        l.s     $f2,36($fp)
        l.s     $f0,36($fp)
        mul.s   $f0,$f2,$f0
        s.s     $f0,200($fp)
        l.s     $f2,196($fp)
        l.s     $f0,200($fp)
        add.s   $f2,$f2,$f0
        l.s     $f0,$LC18
        c.lt.s  $f0,$f2
        bc1t    $L50
        l.s     $f0,32($fp)
        cvt.d.s $f0,$f0
        add.d   $f2,$f0,$f0
        l.s     $f0,36($fp)
        cvt.d.s $f0,$f0
        mul.d   $f0,$f2,$f0
        cvt.s.d $f0,$f0
        s.s     $f0,204($fp)
        l.s     $f2,196($fp)
        l.s     $f0,200($fp)
        sub.s   $f2,$f2,$f0
        l.s     $f0,260($fp)
        add.s   $f0,$f2,$f0
        s.s     $f0,32($fp)
        l.s     $f2,204($fp)
        l.s     $f0,264($fp)
        add.s   $f0,$f2,$f0
        s.s     $f0,36($fp)
        lw      $2,192($fp)
        addu    $2,$2,1
        sw      $2,192($fp)
        b       $L49
$L50:
        addu    $2,$fp,208
        move    $4,$2
        la      $5,$LC19
        lw      $6,192($fp)
        la      $25,sprintf
        jal     $31,$25
        addu    $2,$fp,208
        move    $4,$2
        lw      $5,240($fp)
        la      $25,fputs
        jal     $31,$25
        l.s     $f2,188($fp)
        l.s     $f0,56($fp)
        add.s   $f0,$f2,$f0
        s.s     $f0,188($fp)
        lw      $2,184($fp)
        addu    $2,$2,1
        sw      $2,184($fp)
        b       $L45
$L46:
        la      $4,$LC20
        lw      $5,240($fp)
        la      $25,fputs
        jal     $31,$25
        l.s     $f2,176($fp)
        l.s     $f0,60($fp)
        add.s   $f0,$f2,$f0
        s.s     $f0,176($fp)
        lw      $2,180($fp)
        addu    $2,$2,1
        sw      $2,180($fp)
        b       $L41
$L40:
        move    $sp,$fp
        lw      $31,232($sp)
        lw      $fp,228($sp)
        addu    $sp,$sp,240
        j       $31
        .end    drawJuliaSet
        .size   drawJuliaSet, .-drawJuliaSet
        .align  2
        .globl  getDelimSymbol
        .ent    getDelimSymbol
getDelimSymbol:
        .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)
        lw      $4,48($fp)
        la      $25,strlen
        jal     $31,$25
        sw      $2,24($fp)
$L56:
        lw      $2,24($fp)
        bgtz    $2,$L58
        b       $L57
$L58:
        lw      $3,48($fp)
        lw      $2,24($fp)
        addu    $2,$3,$2
        addu    $2,$2,-1
        lbu     $2,0($2)
        sb      $2,29($fp)
        lb      $3,29($fp)
        lw      $2,_ctype_
        addu    $2,$3,$2
        addu    $2,$2,1
        lbu     $2,0($2)
        srl     $2,$2,2
        andi    $2,$2,0x1
        bne     $2,$0,$L59
        lb      $3,29($fp)
        li      $2,46                   # 0x2e
        beq     $3,$2,$L59
        lb      $3,29($fp)
        lw      $2,_ctype_
        addu    $2,$3,$2
        addu    $2,$2,1
        lbu     $2,0($2)
        andi    $2,$2,0x3
        bne     $2,$0,$L59
        lbu     $2,29($fp)
        sb      $2,28($fp)
        b       $L57
$L59:
        lw      $2,24($fp)
        addu    $2,$2,-1
        sw      $2,24($fp)
        b       $L56
$L57:
        lb      $2,28($fp)
        move    $sp,$fp
        lw      $31,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        j       $31
        .end    getDelimSymbol
        .size   getDelimSymbol, .-getDelimSymbol
        .rdata
        .align  2
$LC21:
        .ascii  "\000"
        .space  9
        .align  2
$LC22:
        .word   -1082130432
        .text
        .align  2
        .globl  getReValue
        .ent    getReValue
getReValue:
        .frame  $fp,80,$31              # vars= 40, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $25
        .set    reorder
        subu    $sp,$sp,80
        .cprestore 16
        sw      $31,72($sp)
        sw      $fp,68($sp)
        sw      $28,64($sp)
        move    $fp,$sp
        sw      $4,80($fp)
        move    $2,$5
        sb      $2,24($fp)
        lbu     $2,$LC21
        sb      $2,32($fp)
        sb      $0,33($fp)
        sb      $0,34($fp)
        sb      $0,35($fp)
        sb      $0,36($fp)
        sb      $0,37($fp)
```

```
        sb      $0,38($fp)
        sb      $0,39($fp)
        sb      $0,40($fp)
        sb      $0,41($fp)
        sw      $0,48($fp)
$L61:
        lw      $4,80($fp)
        la      $25,strlen
        jal     $31,$25
        lw      $3,48($fp)
        sltu    $2,$3,$2
        bne     $2,$0,$L63
        b       $L62
$L63:
        lw      $3,80($fp)
        lw      $2,48($fp)
        addu    $2,$3,$2
        lbu     $2,0($2)
        sb      $2,52($fp)
        lw      $2,48($fp)
        blez    $2,$L64
        lb      $3,52($fp)
        lb      $2,24($fp)
        bne     $3,$2,$L64
        b       $L62
$L64:
        lb      $3,52($fp)
        lw      $2,__ctype_
        addu    $2,$3,$2
        addu    $2,$2,1
        lbu     $2,0($2)
        srl     $2,$2,2
        andi    $2,$2,0x1
        bne     $2,$0,$L67
        lb      $3,52($fp)
        li      $2,46                   # 0x2e
        beq     $3,$2,$L67
        b       $L66
$L67:
        addu    $3,$fp,32
        lw      $2,48($fp)
        addu    $3,$3,$2
        lbu     $2,52($fp)
        sb      $2,0($3)
        b       $L65
$L66:
        l.s     $f0,$LC22
        s.s     $f0,56($fp)
        b       $L60
$L65:
        lw      $2,48($fp)
        addu    $2,$2,1
        sw      $2,48($fp)
        b       $L61
$L62:
        addu    $2,$fp,32
        move    $4,$2
        la      $25,atof
        jal     $31,$25
        cvt.s.d $f0,$f0
        s.s     $f0,56($fp)
$L60:
        l.s     $f0,56($fp)
        move    $sp,$fp
        lw      $31,72($sp)
        lw      $fp,68($sp)
        addu    $sp,$sp,80
        j       $31
        .end    getReValue
        .size   getReValue, .-getReValue
        .rdata
        .align  2
$LC23:
        .word   -1082130432
        .text
        .align  2
        .globl  getImValue
        .ent    getImValue
getImValue:
        .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)
        move    $2,$5
        sb      $2,24($fp)
        lw      $4,72($fp)
        la      $25,strlen
        jal     $31,$25
        sw      $2,28($fp)
        lbu     $2,$LC21
        sb      $2,32($fp)
        sb      $0,33($fp)
        sb      $0,34($fp)
        sb      $0,35($fp)
        sb      $0,36($fp)
        sb      $0,37($fp)
        sb      $0,38($fp)
        sb      $0,39($fp)
        sb      $0,40($fp)
        sb      $0,41($fp)
$L70:
        lw      $2,28($fp)
        bgtz    $2,$L72
        b       $L71
$L72:
        lw      $3,72($fp)
        lw      $2,28($fp)
        addu    $2,$3,$2
        addu    $2,$2,-1
        lbu     $2,0($2)
        sb      $2,48($fp)
        lb      $3,48($fp)
        lb      $2,24($fp)
        bne     $3,$2,$L73
        b       $L71
$L73:
        lb      $3,48($fp)
        lw      $2,_ctype_
        addu    $2,$3,$2
        addu    $2,$2,1
        lbu     $2,0($2)
        srl     $2,$2,2
        andi    $2,$2,0x1
        bne     $2,$0,$L76
        lb      $3,48($fp)
        li      $2,46                   # 0x2e
        beq     $3,$2,$L76
        b       $L75
$L76:
        addu    $3,$fp,31
        lw      $2,28($fp)
        addu    $3,$3,$2
        lbu     $2,48($fp)
        sb      $2,0($3)
        b       $L74
$L75:
        l.s     $f0,$LC23
        s.s     $f0,52($fp)
        b       $L69
$L74:
        lw      $2,28($fp)
        addu    $2,$2,-1
        sw      $2,28($fp)
        b       $L70
$L71:
        addu    $3,$fp,32
        lw      $2,28($fp)
        addu    $2,$3,$2
        move    $4,$2
        la      $25,atof
        jal     $31,$25
        cvt.s.d $f0,$f0
        s.s     $f0,52($fp)
$L69:
        l.s     $f0,52($fp)
        move    $sp,$fp
        lw      $31,64($sp)
        lw      $fp,60($sp)
        addu    $sp,$sp,72
        j       $31
        .end    getImValue
        .size   getImValue, .-getImValue
        .rdata
        .align  2
$LC24:
        .ascii  "i\000"
        .text
        .align  2
        .globl  isComplexValue
        .ent    isComplexValue
isComplexValue:
        .frame  $fp,56,$31              # vars= 16, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
```

```
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $25
        .set    reorder
        subu    $sp,$sp,56
        .cprestore 16
        sw      $31,48($sp)
        sw      $fp,44($sp)
        sw      $28,40($sp)
        move    $fp,$sp
        sw      $4,56($fp)
        lw      $4,56($fp)
        la      $25,strlen
        jal     $31,$25
        sw      $2,24($fp)
        lw      $3,56($fp)
        lw      $2,24($fp)
        addu    $2,$3,$2
        addu    $2,$2,-1
        sw      $2,28($fp)
        lw      $4,28($fp)
        la      $5,$LC24
        la      $25,strcmp
        jal     $31,$25
        beq     $2,$0,$L79
        sw      $0,32($fp)
        b       $L78
$L79:
        li      $2,1                    # 0x1
        sw      $2,32($fp)
$L78:
        lw      $2,32($fp)
        move    $sp,$fp
        lw      $31,48($sp)
        lw      $fp,44($sp)
        addu    $sp,$sp,56
        j       $31
        .end    isComplexValue
        .size   isComplexValue, .-isComplexValue
        .rdata
        .align  2
$LC25:
        .ascii  "\000"
        .align  2
$LC26:
        .word   -1082130432
        .text
        .align  2
        .globl  analyzerComplexParameter
        .ent    analyzerComplexParameter
analyzerComplexParameter:
        .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)
        sw      $6,56($fp)
        lw      $4,48($fp)
        la      $5,$LC25
        la      $25,strcmp
        jal     $31,$25
        bne     $2,$0,$L81
        li      $2,-1                   # 0xffffffffffffffff
        sw      $2,28($fp)
        b       $L80
$L81:
        lw      $4,48($fp)
        la      $25,isComplexValue
        jal     $31,$25
        bne     $2,$0,$L82
        li      $2,-1                   # 0xffffffffffffffff
        sw      $2,28($fp)
        b       $L80
$L82:
        lw      $4,48($fp)
        la      $5,$LC24
        la      $25,strtok
        jal     $31,$25
        lw      $4,48($fp)
        la      $25,getDelimSymbol
        jal     $31,$25
        sb      $2,24($fp)
```

```
        lb      $2,24($fp)
        lw      $4,48($fp)
        move    $5,$2
        la      $25,getReValue
        jal     $31,$25
        lw      $2,52($fp)
        s.s     $f0,0($2)
        lb      $2,24($fp)
        lw      $4,48($fp)
        move    $5,$2
        la      $25,getImValue
        jal     $31,$25
        lw      $2,56($fp)
        s.s     $f0,0($2)
        lw      $2,52($fp)
        l.s     $f2,0($2)
        l.s     $f0,$LC26
        c.eq.s  $f2,$f0
        bc1t    $L84
        lw      $2,56($fp)
        l.s     $f2,0($2)
        l.s     $f0,$LC26
        c.eq.s  $f2,$f0
        bc1t    $L84
        b       $L83
$L84:
        li      $2,-1                   # 0xffffffffffffffff
        sw      $2,28($fp)
        b       $L80
$L83:
        sw      $0,28($fp)
$L80:
        lw      $2,28($fp)
        move    $sp,$fp
        lw      $31,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        j       $31
        .end    analyzerComplexParameter
        .size   analyzerComplexParameter, .-analyzerComplexParameter
        .rdata
        .align  2
$LC27:
        .ascii  "x\000"
        .text
        .align  2
        .globl  setResolution
        .ent    setResolution
setResolution:
        .frame  $fp,56,$31              # vars= 16, regs= 4/0, args= 16, extra= 8
        .mask   0xd0010000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $25
        .set    reorder
        subu    $sp,$sp,56
        .cprestore 16
        sw      $31,52($sp)
        sw      $fp,48($sp)
        sw      $28,44($sp)
        sw      $16,40($sp)
        move    $fp,$sp
        sw      $4,56($fp)
        sw      $5,60($fp)
        sw      $6,64($fp)
        la      $2,$LC27
        sw      $2,24($fp)
        lw      $4,56($fp)
        lw      $5,24($fp)
        la      $25,strtok
        jal     $31,$25
        sw      $2,28($fp)
        lw      $16,60($fp)
        lw      $4,28($fp)
        la      $25,atof
        jal     $31,$25
        cvt.s.d $f0,$f0
        s.s     $f0,0($16)
        move    $4,$0
        lw      $5,24($fp)
        la      $25,strtok
        jal     $31,$25
        sw      $2,28($fp)
        lw      $2,28($fp)
        bne     $2,$0,$L86
        li      $2,-1                   # 0xffffffffffffffff
        sw      $2,32($fp)
        b       $L85
$L86:
        lw      $16,64($fp)
        lw      $4,28($fp)
```

```
        la      $25,atof
        jal     $31,$25
        cvt.s.d $f0,$f0
        s.s     $f0,0($16)
        lw      $2,60($fp)
        l.s     $f2,0($2)
        mtc1    $0,$f0
        c.lt.s  $f0,$f2
        bc1t    $L89
        b       $L88
$L89:
        lw      $2,64($fp)
        l.s     $f2,0($2)
        mtc1    $0,$f0
        c.lt.s  $f0,$f2
        bc1t    $L87
$L88:
        li      $2,-1                   # 0xffffffffffffffff
        sw      $2,32($fp)
        b       $L85
$L87:
        sw      $0,32($fp)
$L85:
        lw      $2,32($fp)
        move    $sp,$fp
        lw      $31,52($sp)
        lw      $fp,48($sp)
        lw      $16,40($sp)
        addu    $sp,$sp,56
        j       $31
        .end    setResolution
        .size   setResolution, .-setResolution
        .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

7 Conclusiones

Se implementó un algoritmo para dibujar el conjunto Julia y sus vecindades, variando los parámetros de entrada.
En las imágenes de pruebas observamos que los puntos negros pertenecen al conjunto mientras que los de tonalidades blancos no. Los distintos "blancos" denotan la velocidad con la que diverge la sucesión (su módulo tiende a infinito).
Por otra parte, nos familiarizamos con MIPS y las distintas herramientas necesarias para el desarrollo de los siguientes trabajos.

8 Enunciado

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`.

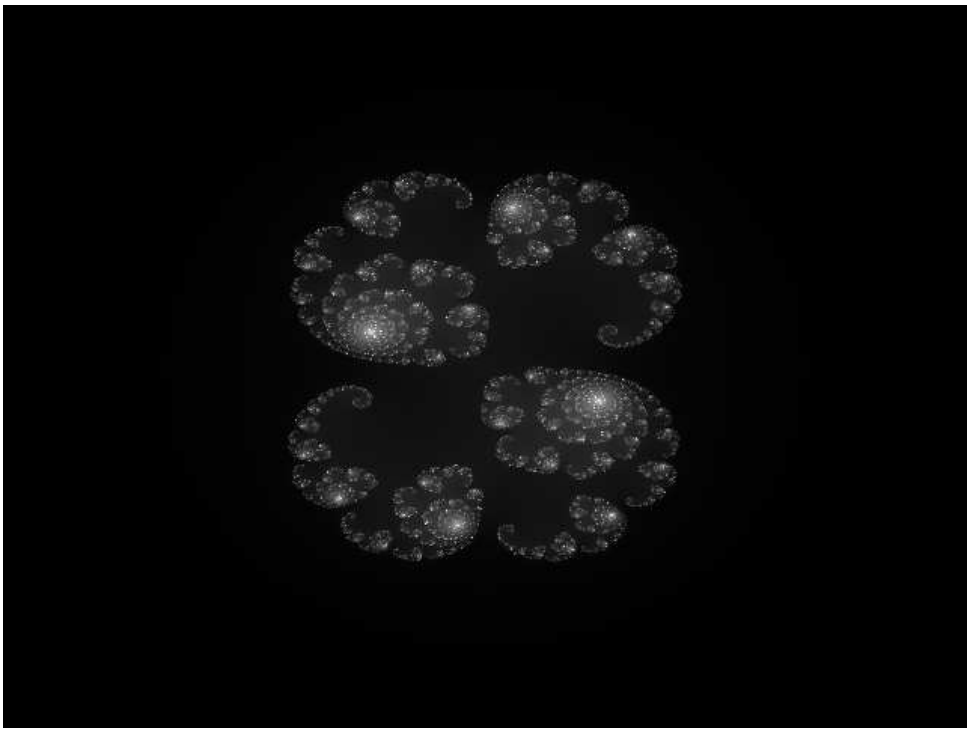

Figura 1: Región barrida por defecto.

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.

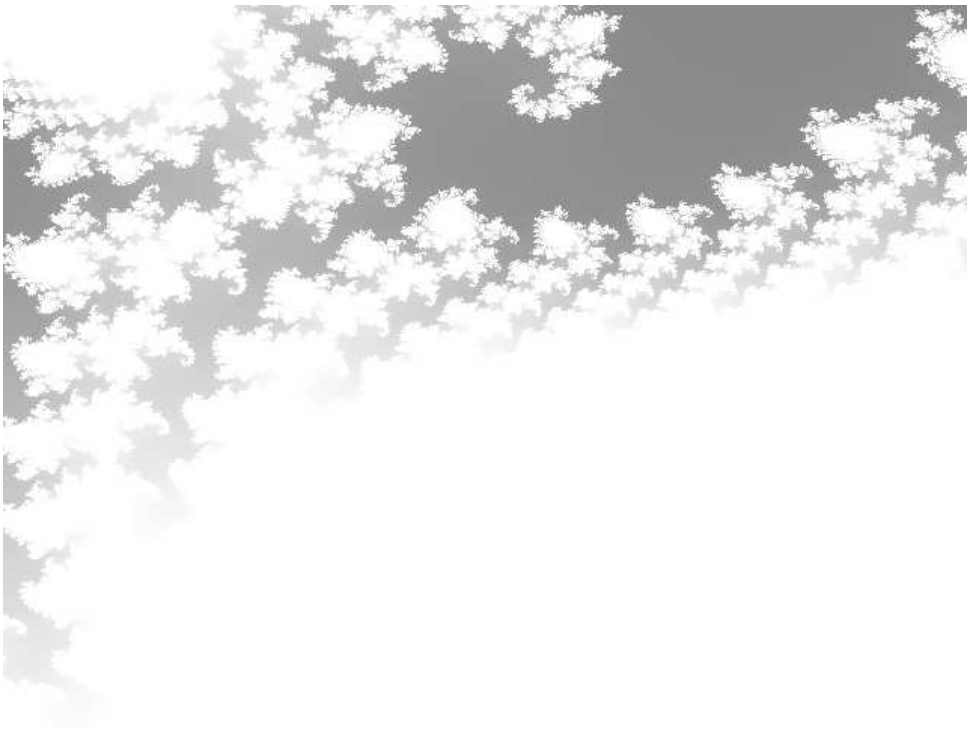

Figura 2: Región comprendida entre $0{,}2795 - 0{,}0075i$ y $0{,}2845 - 0{,}0125i$.

```
$ 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.
```