Universidad De Buenos Aires

Facultad De Ingeniería

66.20 Organización De Computadoras

Trabajo Práctico 0

Integrantes:
Daniel Fernandez - 93083
Nicolas Ortoleva - 93196
Maximiliano Schultheis - 93285

8 de Abril de 2014

Índice

1. Diseño e implementación

El programa posee dos modos implementados: *encode* y *decode*.

El primero es el predeterminado y utiliza un vector (es decir, una tabla) para realizar el pasaje de la representación decimal a hexadecimal. En primer lugar, se obtiene el nibble más significativo del byte leído, se procede al cambio de base de la forma mencionada, se lo imprime por el *output* configurado y luego se procesa de la misma manera el nibble menos significativo correspondiente.

El segundo modo, en cambio, recibe sólo caracteres pertenecientes a la codificación hexadecimal. Por lo tanto, basta con calcular su distancia relativa al cero (como caracter y con el valor indicado por la tabla ASCII) o a la 'A', según sea el caso, para obtener los dos nibbles del byte a decodificar. Una vez hecho esto, se los une mediante la operación lógica 'OR' y se imprime el resultado como en el modo *encode*.

A su vez, la entrada y la salida utilizadas por el programa pueden ser modificadas mediante la incorporación de ciertas opciones particulares en la línea de comandos, logrando así que se tome la estándar provista por el sistema operativo o que se trabaje con archivos elegidos por el usuario.

Para obtener una descripción detallada del modo de uso del programa, se debe ejecutar el comando:

- GuestOS$ ./tp0 -h

2. Comandos de compilación

Sea $TP0DIR el path absoluto en el GuestOS al archivo tp0.c, entonces:

- GuestOS$ cd $TP0DIR
- GuestOS$ gcc -Wall -o tp0 tp0.c

3. Pruebas realizadas

3.1. Primeras pruebas

```
#!/bin/bash

prueba="Archivo vacio"
touch /tmp/zero.txt
./tp0 -a encode -i /tmp/zero.txt -o /tmp/zero.txt.b16
longitud=`ls -la /tmp/zero.txt.b16 | awk '{print $5}'`
if [[ $longitud -eq 0 ]] ; then echo "ok: $prueba"; else echo "ERROR: $prueba" ; fi
rm -f /tmp/zero.txt /tmp/zero.txt.b16

prueba="Codificacion de 'M' por entrada estandar"
hexa=`echo -n M | ./tp0`
if [[ "$hexa" == "4D" ]] ; then echo "ok: $prueba"; else echo "ERROR: $prueba" ; fi

prueba="Codificacion de 'Ma' por entrada estandar"
hexa=`echo -n Ma | ./tp0`
if [[ "$hexa" == "4D61" ]] ; then echo "ok: $prueba"; else echo "ERROR: $prueba" ; fi

prueba="Codificacion de 'Man' por entrada estandar"
hexa=`echo -n Man | ./tp0`
if [[ "$hexa" == "4D616E" ]] ; then echo "ok: $prueba"; else echo "ERROR: $prueba" ;
    fi

prueba="Codificacion y decodificacion de 'Man' por entrada estandar"
mensaje=`echo -n Man | ./tp0 | ./tp0 -a decode`
if [[ "$mensaje" == "Man" ]] ; then echo "ok: $prueba"; else echo "ERROR: $prueba" ;
    fi

prueba="Verificacion bit a bit de codificacion y decodificacion de xyz\n"
```

```
esperado="0000000   x   y   z  \n
0000004"
resultado=`echo xyz | ./tp0 | ./tp0 -a decode | od -t c`
if [[ "$resultado" == "$esperado" ]] ; then echo "ok: $prueba"; else echo "ERROR:
    $prueba" ; fi
```

3.2. Prueba de archivos aleatorios

```
n=1;
while :; do
 head -c $n </dev/urandom >/tmp/in.bin;
 ./tp0 -a encode -i /tmp/in.bin -o /tmp/out.b16;
 ./tp0 -a decode -i /tmp/out.b16 -o /tmp/out.bin;

        if diff /tmp/in.bin /tmp/out.bin; then :; else
                echo ERROR: $n;
                break;

        fi

echo ok: $n;

        n="`expr $n + 1`";

rm -f /tmp/in.bin /tmp/out.b16 /tmp/out.bin

done
```

4. Código Fuente

4.1. Código fuente C

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


static bool encoderActivo = true;

static char vecHexa [] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E'
    ,'F'};

static struct option long_options[] = {
         {"version", no_argument, 0, 'v'},
     {"help", no_argument, 0, 'h'},
     {"input", required_argument, 0, 'i'},
     {"output", required_argument, 0, 'o'},
     {"action", required_argument, 0, 'a'},
     {0, 0, 0, 0}
};


void encoder (char* valorHexa, unsigned int numInt) {
        int highNible = numInt & 0xf0;
```

```
24          int lowNible = numInt & 0x0f;
25          int primerNum = highNible >> 4;
26          int segundoNum = lowNible;
27
28          valorHexa[0] = vecHexa[primerNum];
29          valorHexa[1] = vecHexa[segundoNum];
30          valorHexa[2] = '\0';
31  }
32
33  int correrReferencia ( int numInt ){
34          if ( numInt > 47 && numInt < 58)
35                  return numInt - 48 ;
36
37          else if ( numInt > 64 && numInt < 71)
38                  return numInt - 55 ;
39
40          else if ( numInt > 96 && numInt < 103)
41                  return numInt - 87 ;
42          else {
43                  fprintf(stderr,"Contiene caracteres que no pertenecen al codigo Hexa\
                        n");
44                  exit(1);
45          }
46
47  }
48
49  char decoder (int numPri, int numSeg) {
50          int valor1 = correrReferencia (numPri);
51          int valor2 = correrReferencia (numSeg);
52
53          int highNible = valor1 << 4;
54          highNible =  highNible & 0xf0;
55          valor2 = valor2 & 0x0f;
56
57          char caracter = highNible | valor2;
58
59          return caracter;
60  }
61
62  int leer (FILE* finput) {
63          int caracter = fgetc (finput);
64          if (ferror(finput)) {
65                  fprintf(stderr,"Error al leer el archivo de entrada\n");
66                  exit(1);
67          }
68          return caracter;
69  }
70
71  void escribir_error () {
72          fprintf(stderr,"Error al escribir el archivo de salida\n");
73          exit(1);
74  }
75
76  void procesarArchivos (FILE* finput, FILE* foutput) {
77          int c;
78          int caracter2;
79          int caracter = leer(finput);
80
81          while (caracter != EOF) {
82                  if (encoderActivo) {
83                          char string[3];
```

```
84                          encoder (string, caracter);
85                          if (fputs(string, foutput) == EOF) escribir_error();
86
87                  } else {
88                          caracter2 = leer(finput);
89                          c = decoder(caracter,caracter2);
90                          if (fputc(c, foutput) == EOF) escribir_error();
91                  }
92                  caracter = leer(finput);
93          }
94
95          if (finput != stdin) fclose(finput);
96          if (foutput != stdout) fclose(foutput);
97  }
98
99  void comprobarAction (char* optarg) {
100         if ( strcmp (optarg, "encode") == 0 ) {
101                 encoderActivo = true;
102         }
103         if ( strcmp (optarg, "decode") == 0 ) {
104                 encoderActivo = false;
105         }
106
107 }
108
109 void imprimirAyuda () {
110         printf("Usage:\n");
111         printf("\t ./tp0 -h\n");
112         printf("\t ./tp0 -v\n");
113         printf("Options:\n");
114         printf("\t -v, --version, Shows the version of TP. \n");
115         printf("\t -h, --help , Show help \n");
116         printf("\t -i, --input, Location of the input file\n");
117         printf("\t -o, --output, Location of the output file\n");
118         printf("\t -a, --action, Program action: encode (default) or decode \n");
119         printf("Example: \n");
120         printf("\t ./tp0 -a encode -i /input -o /output -h\n");
121         printf("\t ./tp0 -a decode\n");
122 }
123
124 int opciones (int argc , char** argv, FILE** finput, FILE** foutput) {
125
126         int option_index = 0;
127         int option = getopt_long ( argc, argv, "vhi:o:a:", long_options, &
                option_index);
128         while ( option != -1 ) {
129
130                 switch (option) {
131                         case 'v':
132                                         printf("66.20-Organizacion de Computadoras TP
                                                Version 0.0\n");
133                                         return 1;
134                                         break;
135                         case 'h':
136                                         imprimirAyuda();
137                                         return 1;
138                                         break;
139                         case 'i':
140                                         (*finput) = fopen(optarg,"r");
141                                         if ((*finput) == NULL) {
```

```
                                                fprintf(stderr,"Error al abrir el
                                                    archivo input %s\n",optarg);
                                                exit(1);
                                        }
                                        break;
                                case 'o':
                                                (*foutput) = fopen(optarg, "w");
                                                if ((*foutput) == NULL) {
                                                fprintf(stderr,"Error al abrir el
                                                    archivo output %s \n",optarg);
                                                exit(1);
                                                }
                                                break;
                                case 'a':
                                                comprobarAction(optarg);
                                                break;
                                default:
                                                break;
                        }

                        option = getopt_long ( argc, argv, "vhi:o:a:", long_options, &
                            option_index);
                }

                return 0;
        }

        int main (int argc, char** argv) {
                FILE* finput = stdin;
                FILE* foutput = stdout;
                int opcion = opciones (argc, argv, &finput, &foutput);

                if (opcion == 0) procesarArchivos (finput, foutput);
                return 0;
        }
```

4.2. Código assembly MIPS

```
        .file   1 "tp0.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .data
        .type   encoderActivo, @object
        .size   encoderActivo, 1
encoderActivo:
        .byte   1
        .align  2
        .type   vecHexa, @object
        .size   vecHexa, 16
vecHexa:
        .byte   48
        .byte   49
        .byte   50
        .byte   51
        .byte   52
        .byte   53
        .byte   54
        .byte   55
```

```
        .byte   56
        .byte   57
        .byte   65
        .byte   66
        .byte   67
        .byte   68
        .byte   69
        .byte   70
        .rdata
        .align  2
$LC0:
        .ascii  "version\000"
        .align  2
$LC1:
        .ascii  "help\000"
        .align  2
$LC2:
        .ascii  "input\000"
        .align  2
$LC3:
        .ascii  "output\000"
        .align  2
$LC4:
        .ascii  "action\000"
        .data
        .align  2
        .type   long_options, @object
        .size   long_options, 96
long_options:
        .word   $LC0
        .word   0
        .word   0
        .word   118
        .word   $LC1
        .word   0
        .word   0
        .word   104
        .word   $LC2
        .word   1
        .word   0
        .word   105
        .word   $LC3
        .word   1
        .word   0
        .word   111
        .word   $LC4
        .word   1
        .word   0
        .word   97
        .word   0
        .word   0
        .word   0
        .word   0
        .text
        .align  2
        .globl  encoder
        .ent    encoder
encoder:
        .frame  $fp,32,$ra              # vars= 16, regs= 2/0, args= 0, extra= 8
        .mask   0x50000000,-4
        .fmask  0x00000000,0
```

```
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,32
        .cprestore 0
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,32($fp)
        sw      $a1,36($fp)
        lw      $v0,36($fp)
        andi    $v0,$v0,0xf0
        sw      $v0,8($fp)
        lw      $v0,36($fp)
        andi    $v0,$v0,0xf
        sw      $v0,12($fp)
        lw      $v0,8($fp)
        sra     $v0,$v0,4
        sw      $v0,16($fp)
        lw      $v0,12($fp)
        sw      $v0,20($fp)
        lw      $a0,32($fp)
        lw      $v1,16($fp)
        la      $v0,vecHexa
        addu    $v0,$v1,$v0
        lbu     $v0,0($v0)
        sb      $v0,0($a0)
        lw      $v0,32($fp)
        addu    $a0,$v0,1
        lw      $v1,20($fp)
        la      $v0,vecHexa
        addu    $v0,$v1,$v0
        lbu     $v0,0($v0)
        sb      $v0,0($a0)
        lw      $v0,32($fp)
        addu    $v0,$v0,2
        sb      $zero,0($v0)
        move    $sp,$fp
        lw      $fp,28($sp)
        addu    $sp,$sp,32
        j       $ra
        .end    encoder
        .size   encoder, .-encoder
        .rdata
        .align  2
$LC5:
        .ascii  "Contiene caracteres que no pertenecen al codigo Hexa\n\000"
        .text
        .align  2
        .globl  correrReferencia
        .ent    correrReferencia
correrReferencia:
        .frame  $fp,48,$ra              # vars= 8, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,48
        .cprestore 16
        sw      $ra,40($sp)
```

```
144           sw      $fp,36($sp)
145           sw      $gp,32($sp)
146           move    $fp,$sp
147           sw      $a0,48($fp)
148           lw      $v0,48($fp)
149           slt     $v0,$v0,48
150           bne     $v0,$zero,$L19
151           lw      $v0,48($fp)
152           slt     $v0,$v0,58
153           beq     $v0,$zero,$L19
154           lw      $v0,48($fp)
155           addu    $v0,$v0,-48
156           sw      $v0,24($fp)
157           b       $L18
158   $L19:
159           lw      $v0,48($fp)
160           slt     $v0,$v0,65
161           bne     $v0,$zero,$L21
162           lw      $v0,48($fp)
163           slt     $v0,$v0,71
164           beq     $v0,$zero,$L21
165           lw      $v0,48($fp)
166           addu    $v0,$v0,-55
167           sw      $v0,24($fp)
168           b       $L18
169   $L21:
170           lw      $v0,48($fp)
171           slt     $v0,$v0,97
172           bne     $v0,$zero,$L23
173           lw      $v0,48($fp)
174           slt     $v0,$v0,103
175           beq     $v0,$zero,$L23
176           lw      $v0,48($fp)
177           addu    $v0,$v0,-87
178           sw      $v0,24($fp)
179           b       $L18
180   $L23:
181           la      $a0,__sF+176
182           la      $a1,$LC5
183           la      $t9,fprintf
184           jal     $ra,$t9
185           li      $a0,1                   # 0x1
186           la      $t9,exit
187           jal     $ra,$t9
188   $L18:
189           lw      $v0,24($fp)
190           move    $sp,$fp
191           lw      $ra,40($sp)
192           lw      $fp,36($sp)
193           addu    $sp,$sp,48
194           j       $ra
195           .end    correrReferencia
196           .size   correrReferencia, .-correrReferencia
197           .align  2
198           .globl  decoder
199           .ent    decoder
200   decoder:
201           .frame  $fp,56,$ra              # vars= 16, regs= 3/0, args= 16, extra= 8
202           .mask   0xd0000000,-8
203           .fmask  0x00000000,0
204           .set    noreorder
```

```
205           .cpload $t9
206           .set    reorder
207           subu    $sp,$sp,56
208           .cprestore 16
209           sw      $ra,48($sp)
210           sw      $fp,44($sp)
211           sw      $gp,40($sp)
212           move    $fp,$sp
213           sw      $a0,56($fp)
214           sw      $a1,60($fp)
215           lw      $a0,56($fp)
216           la      $t9,correrReferencia
217           jal     $ra,$t9
218           sw      $v0,24($fp)
219           lw      $a0,60($fp)
220           la      $t9,correrReferencia
221           jal     $ra,$t9
222           sw      $v0,28($fp)
223           lw      $v0,24($fp)
224           sll     $v0,$v0,4
225           sw      $v0,32($fp)
226           lw      $v0,32($fp)
227           andi    $v0,$v0,0xf0
228           sw      $v0,32($fp)
229           lw      $v0,28($fp)
230           andi    $v0,$v0,0xf
231           sw      $v0,28($fp)
232           lbu     $v1,32($fp)
233           lbu     $v0,28($fp)
234           or      $v0,$v1,$v0
235           sb      $v0,36($fp)
236           lb      $v0,36($fp)
237           move    $sp,$fp
238           lw      $ra,48($sp)
239           lw      $fp,44($sp)
240           addu    $sp,$sp,56
241           j       $ra
242           .end    decoder
243           .size   decoder, .-decoder
244           .rdata
245           .align  2
246   $LC6:
247           .ascii  "Error al leer el archivo de entrada\n\000"
248           .text
249           .align  2
250           .globl  leer
251           .ent    leer
252   leer:
253           .frame  $fp,48,$ra              # vars= 8, regs= 3/0, args= 16, extra= 8
254           .mask   0xd0000000,-8
255           .fmask  0x00000000,0
256           .set    noreorder
257           .cpload $t9
258           .set    reorder
259           subu    $sp,$sp,48
260           .cprestore 16
261           sw      $ra,40($sp)
262           sw      $fp,36($sp)
263           sw      $gp,32($sp)
264           move    $fp,$sp
265           sw      $a0,48($fp)
```

```
        lw      $a0,48($fp)
        la      $t9,fgetc
        jal     $ra,$t9
        sw      $v0,24($fp)
        lw      $v0,48($fp)
        lhu     $v0,12($v0)
        srl     $v0,$v0,6
        andi    $v0,$v0,0x1
        beq     $v0,$zero,$L27
        la      $a0,__sF+176
        la      $a1,$LC6
        la      $t9,fprintf
        jal     $ra,$t9
        li      $a0,1                   # 0x1
        la      $t9,exit
        jal     $ra,$t9
$L27:
        lw      $v0,24($fp)
        move    $sp,$fp
        lw      $ra,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        j       $ra
        .end    leer
        .size   leer, .-leer
        .rdata
        .align  2
$LC7:
        .ascii  "Error al escribir el archivo de salida\n\000"
        .text
        .align  2
        .globl  escribir_error
        .ent    escribir_error
escribir_error:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        la      $a0,__sF+176
        la      $a1,$LC7
        la      $t9,fprintf
        jal     $ra,$t9
        li      $a0,1                   # 0x1
        la      $t9,exit
        jal     $ra,$t9
        .end    escribir_error
        .size   escribir_error, .-escribir_error
        .align  2
        .globl  procesarArchivos
        .ent    procesarArchivos
procesarArchivos:
        .frame  $fp,64,$ra              # vars= 24, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
```

```
327           .fmask  0x00000000,0
328           .set    noreorder
329           .cpload $t9
330           .set    reorder
331           subu    $sp,$sp,64
332           .cprestore 16
333           sw      $ra,56($sp)
334           sw      $fp,52($sp)
335           sw      $gp,48($sp)
336           move    $fp,$sp
337           sw      $a0,64($fp)
338           sw      $a1,68($fp)
339           lw      $a0,64($fp)
340           la      $t9,leer
341           jal     $ra,$t9
342           sw      $v0,32($fp)
343   $L30:
344           lw      $v1,32($fp)
345           li      $v0,-1                  # 0xffffffffffffffff
346           bne     $v1,$v0,$L32
347           b       $L31
348   $L32:
349           lbu     $v0,encoderActivo
350           beq     $v0,$zero,$L33
351           addu    $v0,$fp,40
352           move    $a0,$v0
353           lw      $a1,32($fp)
354           la      $t9,encoder
355           jal     $ra,$t9
356           addu    $v0,$fp,40
357           move    $a0,$v0
358           lw      $a1,68($fp)
359           la      $t9,fputs
360           jal     $ra,$t9
361           move    $v1,$v0
362           li      $v0,-1                  # 0xffffffffffffffff
363           bne     $v1,$v0,$L35
364           la      $t9,escribir_error
365           jal     $ra,$t9
366           b       $L35
367   $L33:
368           lw      $a0,64($fp)
369           la      $t9,leer
370           jal     $ra,$t9
371           sw      $v0,28($fp)
372           lw      $a0,32($fp)
373           lw      $a1,28($fp)
374           la      $t9,decoder
375           jal     $ra,$t9
376           sw      $v0,24($fp)
377           lw      $a0,24($fp)
378           lw      $a1,68($fp)
379           la      $t9,fputc
380           jal     $ra,$t9
381           move    $v1,$v0
382           li      $v0,-1                  # 0xffffffffffffffff
383           bne     $v1,$v0,$L35
384           la      $t9,escribir_error
385           jal     $ra,$t9
386   $L35:
387           lw      $a0,64($fp)
```

```
388           la      $t9,leer
389           jal     $ra,$t9
390           sw      $v0,32($fp)
391           b       $L30
392   $L31:
393           lw      $v1,64($fp)
394           la      $v0,__sF
395           beq     $v1,$v0,$L37
396           lw      $a0,64($fp)
397           la      $t9,fclose
398           jal     $ra,$t9
399   $L37:
400           lw      $v1,68($fp)
401           la      $v0,__sF+88
402           beq     $v1,$v0,$L29
403           lw      $a0,68($fp)
404           la      $t9,fclose
405           jal     $ra,$t9
406   $L29:
407           move    $sp,$fp
408           lw      $ra,56($sp)
409           lw      $fp,52($sp)
410           addu    $sp,$sp,64
411           j       $ra
412           .end    procesarArchivos
413           .size   procesarArchivos, .-procesarArchivos
414           .rdata
415           .align  2
416   $LC8:
417           .ascii  "encode\000"
418           .align  2
419   $LC9:
420           .ascii  "decode\000"
421           .text
422           .align  2
423           .globl  comprobarAction
424           .ent    comprobarAction
425   comprobarAction:
426           .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16, extra= 8
427           .mask   0xd0000000,-8
428           .fmask  0x00000000,0
429           .set    noreorder
430           .cpload $t9
431           .set    reorder
432           subu    $sp,$sp,40
433           .cprestore 16
434           sw      $ra,32($sp)
435           sw      $fp,28($sp)
436           sw      $gp,24($sp)
437           move    $fp,$sp
438           sw      $a0,40($fp)
439           lw      $a0,40($fp)
440           la      $a1,$LC8
441           la      $t9,strcmp
442           jal     $ra,$t9
443           bne     $v0,$zero,$L40
444           li      $v0,1                   # 0x1
445           sb      $v0,encoderActivo
446   $L40:
447           lw      $a0,40($fp)
448           la      $a1,$LC9
```

```
449         la      $t9,strcmp
450         jal     $ra,$t9
451         bne     $v0,$zero,$L39
452         sb      $zero,encoderActivo
453 $L39:
454         move    $sp,$fp
455         lw      $ra,32($sp)
456         lw      $fp,28($sp)
457         addu    $sp,$sp,40
458         j       $ra
459         .end    comprobarAction
460         .size   comprobarAction, .-comprobarAction
461         .rdata
462         .align  2
463 $LC10:
464         .ascii  "Usage:\n\000"
465         .align  2
466 $LC11:
467         .ascii  "\t ./tp0 -h\n\000"
468         .align  2
469 $LC12:
470         .ascii  "\t ./tp0 -v\n\000"
471         .align  2
472 $LC13:
473         .ascii  "Options:\n\000"
474         .align  2
475 $LC14:
476         .ascii  "\t -v, --version, Shows the version of TP. \n\000"
477         .align  2
478 $LC15:
479         .ascii  "\t -h, --help , Show help \n\000"
480         .align  2
481 $LC16:
482         .ascii  "\t -i, --input, Location of the input file\n\000"
483         .align  2
484 $LC17:
485         .ascii  "\t -o, --output, Location of the output file\n\000"
486         .align  2
487 $LC18:
488         .ascii  "\t -a, --action, Program action: encode (default) or dec"
489         .ascii  "ode \n\000"
490         .align  2
491 $LC19:
492         .ascii  "Example: \n\000"
493         .align  2
494 $LC20:
495         .ascii  "\t ./tp0 -a encode -i /input -o /output -h\n\000"
496         .align  2
497 $LC21:
498         .ascii  "\t ./tp0 -a decode\n\000"
499         .text
500         .align  2
501         .globl  imprimirAyuda
502         .ent    imprimirAyuda
503 imprimirAyuda:
504         .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16, extra= 8
505         .mask   0xd0000000,-8
506         .fmask  0x00000000,0
507         .set    noreorder
508         .cpload $t9
509         .set    reorder
```

```
510         subu    $sp,$sp,40
511         .cprestore 16
512         sw      $ra,32($sp)
513         sw      $fp,28($sp)
514         sw      $gp,24($sp)
515         move    $fp,$sp
516         la      $a0,$LC10
517         la      $t9,printf
518         jal     $ra,$t9
519         la      $a0,$LC11
520         la      $t9,printf
521         jal     $ra,$t9
522         la      $a0,$LC12
523         la      $t9,printf
524         jal     $ra,$t9
525         la      $a0,$LC13
526         la      $t9,printf
527         jal     $ra,$t9
528         la      $a0,$LC14
529         la      $t9,printf
530         jal     $ra,$t9
531         la      $a0,$LC15
532         la      $t9,printf
533         jal     $ra,$t9
534         la      $a0,$LC16
535         la      $t9,printf
536         jal     $ra,$t9
537         la      $a0,$LC17
538         la      $t9,printf
539         jal     $ra,$t9
540         la      $a0,$LC18
541         la      $t9,printf
542         jal     $ra,$t9
543         la      $a0,$LC19
544         la      $t9,printf
545         jal     $ra,$t9
546         la      $a0,$LC20
547         la      $t9,printf
548         jal     $ra,$t9
549         la      $a0,$LC21
550         la      $t9,printf
551         jal     $ra,$t9
552         move    $sp,$fp
553         lw      $ra,32($sp)
554         lw      $fp,28($sp)
555         addu    $sp,$sp,40
556         j       $ra
557         .end    imprimirAyuda
558         .size   imprimirAyuda, .-imprimirAyuda
559         .rdata
560         .align  2
561 $LC22:
562         .ascii  "vhi:o:a:\000"
563         .align  2
564 $LC23:
565         .ascii  "66.20-Organizacion de Computadoras TP Version 0.0\n\000"
566         .align  2
567 $LC24:
568         .ascii  "r\000"
569         .align  2
570 $LC25:
```

```
571           .ascii  "Error al abrir el archivo input %s\n\000"
572           .align  2
573   $LC26:
574           .ascii  "w\000"
575           .align  2
576   $LC27:
577           .ascii  "Error al abrir el archivo output %s \n\000"
578           .text
579           .align  2
580           .globl  opciones
581           .ent    opciones
582   opciones:
583           .frame  $fp,64,$ra              # vars= 16, regs= 3/0, args= 24, extra= 8
584           .mask   0xd0000000,-8
585           .fmask  0x00000000,0
586           .set    noreorder
587           .cpload $t9
588           .set    reorder
589           subu    $sp,$sp,64
590           .cprestore 24
591           sw      $ra,56($sp)
592           sw      $fp,52($sp)
593           sw      $gp,48($sp)
594           move    $fp,$sp
595           sw      $a0,64($fp)
596           sw      $a1,68($fp)
597           sw      $a2,72($fp)
598           sw      $a3,76($fp)
599           sw      $zero,32($fp)
600           addu    $v0,$fp,32
601           sw      $v0,16($sp)
602           lw      $a0,64($fp)
603           lw      $a1,68($fp)
604           la      $a2,$LC22
605           la      $a3,long_options
606           la      $t9,getopt_long
607           jal     $ra,$t9
608           sw      $v0,36($fp)
609   $L44:
610           lw      $v1,36($fp)
611           li      $v0,-1                  # 0xffffffffffffffff
612           bne     $v1,$v0,$L46
613           b       $L45
614   $L46:
615           lw      $v0,36($fp)
616           addu    $v0,$v0,-97
617           sw      $v0,44($fp)
618           lw      $v1,44($fp)
619           sltu    $v0,$v1,22
620           beq     $v0,$zero,$L47
621           lw      $v0,44($fp)
622           sll     $v1,$v0,2
623           la      $v0,$L56
624           addu    $v0,$v1,$v0
625           lw      $v0,0($v0)
626           .cpadd  $v0
627           j       $v0
628           .rdata
629           .align  2
630   $L56:
631           .gpword $L54
```

```
632           .gpword $L47
633           .gpword $L47
634           .gpword $L47
635           .gpword $L47
636           .gpword $L47
637           .gpword $L47
638           .gpword $L49
639           .gpword $L50
640           .gpword $L47
641           .gpword $L47
642           .gpword $L47
643           .gpword $L47
644           .gpword $L47
645           .gpword $L52
646           .gpword $L47
647           .gpword $L47
648           .gpword $L47
649           .gpword $L47
650           .gpword $L47
651           .gpword $L47
652           .gpword $L48
653           .text
654   $L48:
655           la      $a0,$LC23
656           la      $t9,printf
657           jal     $ra,$t9
658           li      $v0,1                   # 0x1
659           sw      $v0,40($fp)
660           b       $L43
661   $L49:
662           la      $t9,imprimirAyuda
663           jal     $ra,$t9
664           li      $v1,1                   # 0x1
665           sw      $v1,40($fp)
666           b       $L43
667   $L50:
668           lw      $a0,optarg
669           la      $a1,$LC24
670           la      $t9,fopen
671           jal     $ra,$t9
672           move    $v1,$v0
673           lw      $v0,72($fp)
674           sw      $v1,0($v0)
675           lw      $v0,72($fp)
676           lw      $v0,0($v0)
677           bne     $v0,$zero,$L47
678           la      $a0,__sF+176
679           la      $a1,$LC25
680           lw      $a2,optarg
681           la      $t9,fprintf
682           jal     $ra,$t9
683           li      $a0,1                   # 0x1
684           la      $t9,exit
685           jal     $ra,$t9
686   $L52:
687           lw      $a0,optarg
688           la      $a1,$LC26
689           la      $t9,fopen
690           jal     $ra,$t9
691           move    $v1,$v0
692           lw      $v0,76($fp)
```

```
693           sw      $v1,0($v0)
694           lw      $v0,76($fp)
695           lw      $v0,0($v0)
696           bne     $v0,$zero,$L47
697           la      $a0,__sF+176
698           la      $a1,$LC27
699           lw      $a2,optarg
700           la      $t9,fprintf
701           jal     $ra,$t9
702           li      $a0,1                   # 0x1
703           la      $t9,exit
704           jal     $ra,$t9
705   $L54:
706           lw      $a0,optarg
707           la      $t9,comprobarAction
708           jal     $ra,$t9
709   $L47:
710           addu    $v0,$fp,32
711           sw      $v0,16($sp)
712           lw      $a0,64($fp)
713           lw      $a1,68($fp)
714           la      $a2,$LC22
715           la      $a3,long_options
716           la      $t9,getopt_long
717           jal     $ra,$t9
718           sw      $v0,36($fp)
719           b       $L44
720   $L45:
721           sw      $zero,40($fp)
722   $L43:
723           lw      $v0,40($fp)
724           move    $sp,$fp
725           lw      $ra,56($sp)
726           lw      $fp,52($sp)
727           addu    $sp,$sp,64
728           j       $ra
729           .end    opciones
730           .size   opciones, .-opciones
731           .align  2
732           .globl  main
733           .ent    main
734   main:
735           .frame  $fp,56,$ra              # vars= 16, regs= 3/0, args= 16, extra= 8
736           .mask   0xd0000000,-8
737           .fmask  0x00000000,0
738           .set    noreorder
739           .cpload $t9
740           .set    reorder
741           subu    $sp,$sp,56
742           .cprestore 16
743           sw      $ra,48($sp)
744           sw      $fp,44($sp)
745           sw      $gp,40($sp)
746           move    $fp,$sp
747           sw      $a0,56($fp)
748           sw      $a1,60($fp)
749           la      $v0,__sF
750           sw      $v0,24($fp)
751           la      $v0,__sF+88
752           sw      $v0,28($fp)
753           addu    $v0,$fp,28
```

```
754             lw      $a0,56($fp)
755             lw      $a1,60($fp)
756             addu    $a2,$fp,24
757             move    $a3,$v0
758             la      $t9,opciones
759             jal     $ra,$t9
760             sw      $v0,32($fp)
761             lw      $v0,32($fp)
762             bne     $v0,$zero,$L58
763             lw      $a0,24($fp)
764             lw      $a1,28($fp)
765             la      $t9,procesarArchivos
766             jal     $ra,$t9
767     $L58:
768             move    $v0,$zero
769             move    $sp,$fp
770             lw      $ra,48($sp)
771             lw      $fp,44($sp)
772             addu    $sp,$sp,56
773             j       $ra
774             .end    main
775             .size   main, .-main
776             .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

5. Conclusiones

No sólo se ha logrado compilar y ejecutar un programa en C desde el emulador, comprobando la portabilidad de su código fuente, sino que también se ha notado en términos generales la gran diferencia existente en la velocidad de ejecución entre el anfitrión y el huesped.