

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	2
2. Comandos de compilación	2
3. Pruebas realizadas	2
3.1. Primeras pruebas	2
3.2. Prueba de archivos aleatorios	3
4. Código Fuente	3
4.1. Código fuente C	3
4.2. Código assembly MIPS	6
5. Conclusiones	25

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

```
1  #!/bin/bash
2
3  prueba="Archivo vacio"
4  touch /tmp/zero.txt
5  ./tp0 -a encode -i /tmp/zero.txt -o /tmp/zero.txt.b16
6  longitud='ls -la /tmp/zero.txt.b16 | awk '{print $5}''
7  if [[ $longitud -eq 0 ]] ; then echo "ok: $prueba"; else echo "ERROR: $prueba" ; fi
8  rm -f /tmp/zero.txt /tmp/zero.txt.b16
9
10 prueba="Codificacion de 'M' por entrada estandar"
11 hexa='echo -n M | ./tp0'
12 if [[ "$hexa" == "4D" ]] ; then echo "ok: $prueba"; else echo "ERROR: $prueba" ; fi
13
14 prueba="Codificacion de 'Ma' por entrada estandar"
15 hexa='echo -n Ma | ./tp0'
16 if [[ "$hexa" == "4D61" ]] ; then echo "ok: $prueba"; else echo "ERROR: $prueba" ; fi
17
18 prueba="Codificacion de 'Man' por entrada estandar"
19 hexa='echo -n Man | ./tp0'
20 if [[ "$hexa" == "4D616E" ]] ; then echo "ok: $prueba"; else echo "ERROR: $prueba" ;
    fi
21
22 prueba="Codificacion y decodificacion de 'Man' por entrada estandar"
23 mensaje='echo -n Man | ./tp0 | ./tp0 -a decode'
24 if [[ "$mensaje" == "Man" ]] ; then echo "ok: $prueba"; else echo "ERROR: $prueba" ;
    fi
25
26 prueba="Verificacion bit a bit de codificacion y decodificacion de xyz\n"
```

```

27 esperado="0000000 x y z \n
28 0000004"
29 resultado='echo xyz | ./tp0 | ./tp0 -a decode | od -t c'
30 if [[ "$resultado" == "$esperado" ]] ; then echo "ok: $prueba"; else echo "ERROR:
    $prueba" ; fi

```

3.2. Prueba de archivos aleatorios

```

1 n=1;
2 while ;; do
3     head -c $n </dev/urandom >/tmp/in.bin;
4     ./tp0 -a encode -i /tmp/in.bin -o /tmp/out.b16;
5     ./tp0 -a decode -i /tmp/out.b16 -o /tmp/out.bin;
6
7     if diff /tmp/in.bin /tmp/out.bin; then ;; else
8         echo ERROR: $n;
9         break;
10
11     fi
12
13 echo ok: $n;
14
15 n="'expr $n + 1'";
16
17 rm -f /tmp/in.bin /tmp/out.b16 /tmp/out.bin
18
19 done

```

4. Código Fuente

4.1. Código fuente C

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <getopt.h>
4 #include <string.h>
5 #include <stdbool.h>
6
7
8 static bool encoderActivo = true;
9
10 static char vecHexa [] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E',
    'F'};
11
12 static struct option long_options[] = {
13     {"version", no_argument, 0, 'v'},
14     {"help", no_argument, 0, 'h'},
15     {"input", required_argument, 0, 'i'},
16     {"output", required_argument, 0, 'o'},
17     {"action", required_argument, 0, 'a'},
18     {0, 0, 0, 0}
19 };
20
21
22 void encoder (char* valorHexa, unsigned int numInt) {
23     int highNibble = numInt & 0xf0;

```

```

24     int lowNibble = numInt & 0x0f;
25     int primerNum = highNibble >> 4;
26     int segundoNum = lowNibble;
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\
44             n");
45         exit(1);
46     }
47 }
48
49 char decoder (int numPri, int numSeg) {
50     int valor1 = correrReferencia (numPri);
51     int valor2 = correrReferencia (numSeg);
52
53     int highNibble = valor1 << 4;
54     highNibble = highNibble & 0xf0;
55     valor2 = valor2 & 0x0f;
56
57     char character = highNibble | valor2;
58
59     return character;
60 }
61
62 int leer (FILE* finput) {
63     int character = fgetc (finput);
64     if (ferror(finput)) {
65         fprintf(stderr,"Error al leer el archivo de entrada\n");
66         exit(1);
67     }
68     return character;
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 character2;
79     int character = leer(finput);
80
81     while (character != EOF) {
82         if (encoderActivo) {
83             char string[3];

```

```

84         encoder (string, character);
85         if (fputs(string, foutput) == EOF) escribir_error();
86
87     } else {
88         character2 = leer(fininput);
89         c = decoder(caracter,character2);
90         if (fputc(c, foutput) == EOF) escribir_error();
91     }
92     caracter = leer(fininput);
93 }
94
95 if (fininput != stdin) fclose(fininput);
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** fininput, FILE** foutput) {
125
126     int option_index = 0;
127     int option = getopt_long ( argc, argv, "vhi:o:a:", long_options, &
128         option_index);
129     while ( option != -1 ) {
130
131         switch (option) {
132             case 'v':
133
134                 printf("66.20-Organizacion de Computadoras TP
135                     Version 0.0\n");
136                 return 1;
137                 break;
138
139             case 'h':
140
141                 imprimirAyuda();
142                 return 1;
143                 break;
144
145             case 'i':
146
147                 (*fininput) = fopen(optarg,"r");
148                 if ((*fininput) == NULL) {

```

```

142                                     fprintf(stderr,"Error al abrir el
143                                     archivo input %s\n",optarg);
144                                     exit(1);
145                                     }
146                                     break;
147     case 'o':
148         (*foutput) = fopen(optarg, "w");
149         if ((*foutput) == NULL) {
150             fprintf(stderr,"Error al abrir el
151             archivo output %s \n",optarg);
152             exit(1);
153         }
154         break;
155     case 'a':
156         comprobarAction(optarg);
157         break;
158     default:
159         break;
160 }
161
162     option = getopt_long ( argc, argv, "vhi:o:a:", long_options, &
163     option_index);
164 }
165
166     return 0;
167 }
168
169 int main (int argc, char** argv) {
170     FILE* finput = stdin;
171     FILE* foutput = stdout;
172     int opcion = opciones (argc, argv, &finput, &foutput);
173
174     if (opcion == 0) procesarArchivos (finput, foutput);
175     return 0;
176 }

```

4.2. Código assembly MIPS

```

1      .file      1 "tp0.c"
2      .section   .mdebug.abi32
3      .previous
4      .abicalls
5      .globl     encoderActivo
6      .data
7      .type      encoderActivo, @object
8      .size      encoderActivo, 1
9      encoderActivo:
10     .byte      1
11     .globl     decoderActivo
12     .globl     decoderActivo
13     .section   .bss
14     .align     0
15     .type      decoderActivo, @object
16     .size      decoderActivo, 1
17     decoderActivo:
18     .space     1
19     .globl     finput
20     .globl     finput
21     .align     2

```

```

22         .type    finput, @object
23         .size    finput, 4
24     finput:
25         .space   4
26         .globl   foutput
27         .globl   foutput
28         .align   2
29         .type    foutput, @object
30         .size    foutput, 4
31     foutput:
32         .space   4
33         .rdata
34         .align   2
35     $LC0:
36         .ascii   "0\000"
37         .align   2
38     $LC1:
39         .ascii   "1\000"
40         .align   2
41     $LC2:
42         .ascii   "2\000"
43         .align   2
44     $LC3:
45         .ascii   "3\000"
46         .align   2
47     $LC4:
48         .ascii   "4\000"
49         .align   2
50     $LC5:
51         .ascii   "5\000"
52         .align   2
53     $LC6:
54         .ascii   "6\000"
55         .align   2
56     $LC7:
57         .ascii   "7\000"
58         .align   2
59     $LC8:
60         .ascii   "8\000"
61         .align   2
62     $LC9:
63         .ascii   "9\000"
64         .align   2
65     $LC10:
66         .ascii   "A\000"
67         .align   2
68     $LC11:
69         .ascii   "B\000"
70         .align   2
71     $LC12:
72         .ascii   "C\000"
73         .align   2
74     $LC13:
75         .ascii   "D\000"
76         .align   2
77     $LC14:
78         .ascii   "E\000"
79         .align   2
80     $LC15:
81         .ascii   "F\000"
82         .globl   vecHexa

```



```

83     .data
84     .align 2
85     .type vecHexa, @object
86     .size vecHexa, 64
87 vecHexa:
88     .word $LC0
89     .word $LC1
90     .word $LC2
91     .word $LC3
92     .word $LC4
93     .word $LC5
94     .word $LC6
95     .word $LC7
96     .word $LC8
97     .word $LC9
98     .word $LC10
99     .word $LC11
100    .word $LC12
101    .word $LC13
102    .word $LC14
103    .word $LC15
104    .rdata
105    .align 2
106 $LC16:
107     .ascii "version\000"
108     .align 2
109 $LC17:
110     .ascii "help\000"
111     .align 2
112 $LC18:
113     .ascii "input\000"
114     .align 2
115 $LC19:
116     .ascii "output\000"
117     .align 2
118 $LC20:
119     .ascii "action\000"
120     .data
121     .align 2
122     .type long_options, @object
123     .size long_options, 96
124 long_options:
125     .word $LC16
126     .word 0
127     .word 0
128     .word 118
129     .word $LC17
130     .word 0
131     .word 0
132     .word 104
133     .word $LC18
134     .word 1
135     .word 0
136     .word 105
137     .word $LC19
138     .word 1
139     .word 0
140     .word 111
141     .word $LC20
142     .word 1
143     .word 0

```

```

144     .word    97
145     .word    0
146     .word    0
147     .word    0
148     .word    0
149     .text
150     .align   2
151     .globl   encoder
152     .ent     encoder
153 encoder:
154     .frame    $fp,72,$ra                # vars= 32, regs= 3/0, args= 16, extra= 8
155     .mask     0xd0000000,-8
156     .fmask    0x00000000,0
157     .set      noreorder
158     .cpload   $t9
159     .set      reorder
160     subu      $sp,$sp,72
161     .cprestore 16
162     sw        $ra,64($sp)
163     sw        $fp,60($sp)
164     sw        $gp,56($sp)
165     move      $fp,$sp
166     sw        $a0,72($fp)
167     lw        $v0,72($fp)
168     andi      $v0,$v0,0xf0
169     sw        $v0,24($fp)
170     lw        $v0,72($fp)
171     andi      $v0,$v0,0xf
172     sw        $v0,28($fp)
173     lw        $v0,24($fp)
174     sra       $v0,$v0,4
175     sw        $v0,32($fp)
176     lw        $v0,28($fp)
177     sw        $v0,36($fp)
178     lw        $v0,32($fp)
179     sll       $v1,$v0,2
180     la        $v0,vecHexa
181     addu      $v0,$v1,$v0
182     lw        $v0,0($v0)
183     sw        $v0,40($fp)
184     lw        $v0,36($fp)
185     sll       $v1,$v0,2
186     la        $v0,vecHexa
187     addu      $v0,$v1,$v0
188     lw        $v0,0($v0)
189     sw        $v0,44($fp)
190     li        $a0,3                    # 0x3
191     la        $t9,malloc
192     jal       $ra,$t9
193     sw        $v0,48($fp)
194     lw        $a0,48($fp)
195     lw        $a1,40($fp)
196     la        $t9,strcpy
197     jal       $ra,$t9
198     lw        $a0,48($fp)
199     lw        $a1,44($fp)
200     la        $t9,strcat
201     jal       $ra,$t9
202     lw        $v0,48($fp)
203     move      $sp,$fp
204     lw        $ra,64($sp)

```

```

205         lw      $fp,60($sp)
206         addu    $sp,$sp,72
207         j       $ra
208         .end    encoder
209         .size   encoder, .-encoder
210         .rdata
211         .align  2
212     $LC21:
213         .ascii  "Contiene caracteres que no pertenecen al codigo Hexa\n\000"
214         .text
215         .align  2
216         .globl  correrReferencia
217         .ent    correrReferencia
218     correrReferencia:
219         .frame   $fp,48,$ra                # vars= 8, regs= 3/0, args= 16, extra= 8
220         .mask    0xd0000000,-8
221         .fmask   0x00000000,0
222         .set     noreorder
223         .cpload  $t9
224         .set     reorder
225         subu     $sp,$sp,48
226         .cprestore 16
227         sw      $ra,40($sp)
228         sw      $fp,36($sp)
229         sw      $gp,32($sp)
230         move     $fp,$sp
231         sw      $a0,48($fp)
232         lw      $v0,48($fp)
233         slt     $v0,$v0,48
234         bne     $v0,$zero,$L19
235         lw      $v0,48($fp)
236         slt     $v0,$v0,58
237         beq     $v0,$zero,$L19
238         lw      $v0,48($fp)
239         addu    $v0,$v0,-48
240         sw      $v0,24($fp)
241         b       $L18
242     $L19:
243         lw      $v0,48($fp)
244         slt     $v0,$v0,65
245         bne     $v0,$zero,$L21
246         lw      $v0,48($fp)
247         slt     $v0,$v0,71
248         beq     $v0,$zero,$L21
249         lw      $v0,48($fp)
250         addu    $v0,$v0,-55
251         sw      $v0,24($fp)
252         b       $L18
253     $L21:
254         lw      $v0,48($fp)
255         slt     $v0,$v0,97
256         bne     $v0,$zero,$L23
257         lw      $v0,48($fp)
258         slt     $v0,$v0,103
259         beq     $v0,$zero,$L23
260         lw      $v0,48($fp)
261         addu    $v0,$v0,-87
262         sw      $v0,24($fp)
263         b       $L18
264     $L23:
265         la      $a0, __sF+176

```

```

266     la      $a1,$LC21
267     la      $t9,fprintf
268     jal     $ra,$t9
269     li      $a0,1                # 0x1
270     la      $t9,exit
271     jal     $ra,$t9
272 $L18:
273     lw      $v0,24($fp)
274     move    $sp,$fp
275     lw      $ra,40($sp)
276     lw      $fp,36($sp)
277     addu    $sp,$sp,48
278     j       $ra
279     .end    correrReferencia
280     .size   correrReferencia,.-correrReferencia
281     .align  2
282     .globl  decoder
283     .ent    decoder
284 decoder:
285     .frame  $fp,56,$ra            # vars= 16, regs= 3/0, args= 16, extra= 8
286     .mask   0xd0000000,-8
287     .fmask  0x00000000,0
288     .set    noreorder
289     .cpld   $t9
290     .set    reorder
291     subu    $sp,$sp,56
292     .cprestore 16
293     sw      $ra,48($sp)
294     sw      $fp,44($sp)
295     sw      $gp,40($sp)
296     move    $fp,$sp
297     sw      $a0,56($fp)
298     sw      $a1,60($fp)
299     lw      $a0,56($fp)
300     la      $t9,correrReferencia
301     jal     $ra,$t9
302     sw      $v0,24($fp)
303     lw      $a0,60($fp)
304     la      $t9,correrReferencia
305     jal     $ra,$t9
306     sw      $v0,28($fp)
307     lw      $v0,24($fp)
308     sll     $v0,$v0,4
309     sw      $v0,32($fp)
310     lw      $v0,32($fp)
311     andi    $v0,$v0,0xf0
312     sw      $v0,32($fp)
313     lw      $v0,28($fp)
314     andi    $v0,$v0,0xf
315     sw      $v0,28($fp)
316     lbu     $v1,32($fp)
317     lbu     $v0,28($fp)
318     or      $v0,$v1,$v0
319     sb      $v0,36($fp)
320     lb      $v0,36($fp)
321     move    $sp,$fp
322     lw      $ra,48($sp)
323     lw      $fp,44($sp)
324     addu    $sp,$sp,56
325     j       $ra
326     .end    decoder

```

```

327         .size    decoder, .-decoder
328         .rdata
329         .align   2
330 $LC22:
331         .ascii   "%s\000"
332         .align   2
333 $LC23:
334         .ascii   "%c\000"
335         .text
336         .align   2
337         .globl   procesarArchivos
338         .ent     procesarArchivos
339 procesarArchivos:
340         .frame    $fp,56,$ra                # vars= 16, regs= 3/0, args= 16, extra= 8
341         .mask     0xd0000000,-8
342         .fmask    0x00000000,0
343         .set      noreorder
344         .cpload   $t9
345         .set      reorder
346         subu      $sp,$sp,56
347         .cprestore 16
348         sw        $ra,48($sp)
349         sw        $fp,44($sp)
350         sw        $gp,40($sp)
351         move      $fp,$sp
352         sw        $a0,56($fp)
353         sw        $a1,60($fp)
354         lw        $a0,56($fp)
355         la        $t9,fgetc
356         jal       $ra,$t9
357         sw        $v0,36($fp)
358 $L27:
359         lw        $v1,36($fp)
360         li        $v0,-1                    # 0xffffffffffffffff
361         bne       $v1,$v0,$L29
362         b         $L28
363 $L29:
364         lbu       $v0,encoderActivo
365         beq       $v0,$zero,$L30
366         lw        $a0,36($fp)
367         la        $t9,encoder
368         jal       $ra,$t9
369         sw        $v0,24($fp)
370         lw        $v0,60($fp)
371         beq       $v0,$zero,$L31
372         lw        $a0,24($fp)
373         lw        $a1,60($fp)
374         la        $t9,fputs
375         jal       $ra,$t9
376         b         $L32
377 $L31:
378         la        $a0,$LC22
379         lw        $a1,24($fp)
380         la        $t9,printf
381         jal       $ra,$t9
382 $L32:
383         lw        $a0,24($fp)
384         la        $t9,free
385         jal       $ra,$t9
386         b         $L33
387 $L30:

```

```

388     lw      $a0,56($fp)
389     la      $t9,fgetc
390     jal     $ra,$t9
391     sw      $v0,32($fp)
392     lw      $a0,36($fp)
393     lw      $a1,32($fp)
394     la      $t9,decoder
395     jal     $ra,$t9
396     sb      $v0,28($fp)
397     lw      $v0,60($fp)
398     beq     $v0,$zero,$L34
399     lb      $v0,28($fp)
400     move    $a0,$v0
401     lw      $a1,60($fp)
402     la      $t9,fputc
403     jal     $ra,$t9
404     b       $L33
405 $L34:
406     lb      $v0,28($fp)
407     la      $a0,$LC23
408     move    $a1,$v0
409     la      $t9,printf
410     jal     $ra,$t9
411 $L33:
412     lw      $a0,56($fp)
413     la      $t9,fgetc
414     jal     $ra,$t9
415     sw      $v0,36($fp)
416     b       $L27
417 $L28:
418     lw      $a0,56($fp)
419     la      $t9,fclose
420     jal     $ra,$t9
421     lw      $v0,60($fp)
422     beq     $v0,$zero,$L26
423     lw      $a0,60($fp)
424     la      $t9,fclose
425     jal     $ra,$t9
426 $L26:
427     move    $sp,$fp
428     lw      $ra,48($sp)
429     lw      $fp,44($sp)
430     addu    $sp,$sp,56
431     j       $ra
432     .end    procesarArchivos
433     .size   procesarArchivos,.-procesarArchivos
434     .align  2
435     .globl  EntradaEncoderStandar
436     .ent    EntradaEncoderStandar
437 EntradaEncoderStandar:
438     .frame  $fp,72,$ra          # vars= 32, regs= 3/0, args= 16, extra= 8
439     .mask   0xd0000000,-8
440     .fmask  0x00000000,0
441     .set    noreorder
442     .cpld   $t9
443     .set    reorder
444     subu    $sp,$sp,72
445     .cprestore 16
446     sw      $ra,64($sp)
447     sw      $fp,60($sp)
448     sw      $gp,56($sp)

```

```

449      move    $fp,$sp
450      move    $a0,$zero
451      la      $t9,isatty
452      jal     $ra,$t9
453      sltu    $v0,$zero,$v0
454      sb      $v0,24($fp)
455      lw      $v0,__$F+4
456      addu    $v0,$v0,-1
457      sw      $v0,__$F+4
458      bgez    $v0,$L38
459      la      $a0,__$F
460      la      $t9,__$srget
461      jal     $ra,$t9
462      sw      $v0,40($fp)
463      b       $L39
464  $L38:
465      la      $v0,__$F
466      lw      $v1,0($v0)
467      move    $a0,$v1
468      lbu     $a0,0($a0)
469      sw      $a0,40($fp)
470      addu    $v1,$v1,1
471      sw      $v1,0($v0)
472  $L39:
473      lw      $v0,40($fp)
474      sw      $v0,28($fp)
475      sb      $zero,44($fp)
476      lw      $v1,28($fp)
477      li      $v0,-1                # 0xffffffffffffffff
478      bne     $v1,$v0,$L42
479      lbu     $v0,24($fp)
480      bne     $v0,$zero,$L42
481      b       $L41
482  $L42:
483      lw      $v1,28($fp)
484      li      $v0,10                # 0xa
485      bne     $v1,$v0,$L40
486      lbu     $v0,24($fp)
487      bne     $v0,$zero,$L41
488      b       $L40
489  $L41:
490      li      $v0,1                # 0x1
491      sb      $v0,44($fp)
492  $L40:
493      lbu     $v0,44($fp)
494      sb      $v0,32($fp)
495  $L43:
496      lbu     $v0,32($fp)
497      beq     $v0,$zero,$L45
498      b       $L37
499  $L45:
500      lw      $a0,28($fp)
501      la      $t9,encoder
502      jal     $ra,$t9
503      sw      $v0,36($fp)
504      lw      $v0,foutput
505      beq     $v0,$zero,$L46
506      lw      $a0,foutput
507      la      $a1,$LC22
508      lw      $a2,36($fp)
509      la      $t9,fprintf

```

```

510     jal    $ra,$t9
511     b      $L47
512 $L46:
513     la     $a0,$LC22
514     lw     $a1,36($fp)
515     la     $t9,printf
516     jal    $ra,$t9
517 $L47:
518     lw     $a0,36($fp)
519     la     $t9,free
520     jal    $ra,$t9
521     lw     $v0,__$sF+4
522     addu   $v0,$v0,-1
523     sw     $v0,__$sF+4
524     bgez   $v0,$L48
525     la     $a0,__$sF
526     la     $t9,__$srget
527     jal    $ra,$t9
528     sw     $v0,48($fp)
529     b      $L49
530 $L48:
531     la     $v0,__$sF
532     lw     $v1,0($v0)
533     move   $a0,$v1
534     lbu    $a0,0($a0)
535     sw     $a0,48($fp)
536     addu   $v1,$v1,1
537     sw     $v1,0($v0)
538 $L49:
539     lw     $v0,48($fp)
540     sw     $v0,28($fp)
541     sb     $zero,52($fp)
542     lw     $v1,28($fp)
543     li     $v0,-1                # 0xffffffffffffffff
544     bne    $v1,$v0,$L52
545     lbu    $v0,24($fp)
546     bne    $v0,$zero,$L52
547     b      $L51
548 $L52:
549     lw     $v1,28($fp)
550     li     $v0,10                # 0xa
551     bne    $v1,$v0,$L50
552     lbu    $v0,24($fp)
553     bne    $v0,$zero,$L51
554     b      $L50
555 $L51:
556     li     $v0,1                 # 0x1
557     sb     $v0,52($fp)
558 $L50:
559     lbu    $v0,52($fp)
560     sb     $v0,32($fp)
561     b      $L43
562 $L37:
563     move   $sp,$fp
564     lw     $ra,64($sp)
565     lw     $fp,60($sp)
566     addu   $sp,$sp,72
567     j      $ra
568     .end   EntradaEncoderStandar
569     .size  EntradaEncoderStandar, .-EntradaEncoderStandar
570     .align 2

```



```

571      .globl  EntradaDecoderStandar
572      .ent    EntradaDecoderStandar
573  EntradaDecoderStandar:
574      .frame   $fp,80,$ra          # vars= 40, regs= 3/0, args= 16, extra= 8
575      .mask    0xd0000000,-8
576      .fmask   0x00000000,0
577      .set     noreorder
578      .cpload  $t9
579      .set     reorder
580      subu     $sp,$sp,80
581      .cprestore 16
582      sw       $ra,72($sp)
583      sw       $fp,68($sp)
584      sw       $gp,64($sp)
585      move     $fp,$sp
586      move     $a0,$zero
587      la       $t9,isatty
588      jal      $ra,$t9
589      sltu     $v0,$zero,$v0
590      sb       $v0,24($fp)
591      lw       $v0,__$sF+4
592      addu     $v0,$v0,-1
593      sw       $v0,__$sF+4
594      bgez     $v0,$L54
595      la       $a0,__$sF
596      la       $t9,__$srget
597      jal      $ra,$t9
598      sw       $v0,40($fp)
599      b        $L55
600  $L54:
601      la       $v0,__$sF
602      lw       $v1,0($v0)
603      move     $a0,$v1
604      lbu      $a0,0($a0)
605      sw       $a0,40($fp)
606      addu     $v1,$v1,1
607      sw       $v1,0($v0)
608  $L55:
609      lw       $v0,40($fp)
610      sw       $v0,28($fp)
611      lw       $v0,__$sF+4
612      addu     $v0,$v0,-1
613      sw       $v0,__$sF+4
614      bgez     $v0,$L56
615      la       $a0,__$sF
616      la       $t9,__$srget
617      jal      $ra,$t9
618      sw       $v0,44($fp)
619      b        $L57
620  $L56:
621      la       $v0,__$sF
622      lw       $v1,0($v0)
623      move     $a0,$v1
624      lbu      $a0,0($a0)
625      sw       $a0,44($fp)
626      addu     $v1,$v1,1
627      sw       $v1,0($v0)
628  $L57:
629      lw       $v0,44($fp)
630      sw       $v0,32($fp)
631      sb       $zero,48($fp)

```

```

632      lw      $v1,28($fp)
633      li      $v0,-1                      # 0xffffffffffffffff
634      bne     $v1,$v0,$L60
635      lbu     $v0,24($fp)
636      bne     $v0,$zero,$L60
637      b       $L59
638 $L60:
639      lw      $v1,28($fp)
640      li      $v0,10                      # 0xa
641      bne     $v1,$v0,$L58
642      lbu     $v0,24($fp)
643      bne     $v0,$zero,$L59
644      b       $L58
645 $L59:
646      li      $v0,1                      # 0x1
647      sb      $v0,48($fp)
648 $L58:
649      lbu     $v0,48($fp)
650      sb      $v0,36($fp)
651 $L61:
652      lbu     $v0,36($fp)
653      beq     $v0,$zero,$L63
654      b       $L53
655 $L63:
656      lw      $a0,28($fp)
657      lw      $a1,32($fp)
658      la      $t9,decoder
659      jal     $ra,$t9
660      sb      $v0,37($fp)
661      lw      $v0,foutput
662      beq     $v0,$zero,$L64
663      lb      $v0,37($fp)
664      move    $a0,$v0
665      lw      $a1,foutput
666      la      $t9,fputc
667      jal     $ra,$t9
668      b       $L65
669 $L64:
670      lb      $v0,37($fp)
671      la      $a0,$LC23
672      move    $a1,$v0
673      la      $t9,printf
674      jal     $ra,$t9
675 $L65:
676      lw      $v0,__$SF+4
677      addu    $v0,$v0,-1
678      sw      $v0,__$SF+4
679      bgez    $v0,$L66
680      la      $a0,__$SF
681      la      $t9,__$srget
682      jal     $ra,$t9
683      sw      $v0,52($fp)
684      b       $L67
685 $L66:
686      la      $v0,__$SF
687      lw      $v1,0($v0)
688      move    $a0,$v1
689      lbu     $a0,0($a0)
690      sw      $a0,52($fp)
691      addu    $v1,$v1,1
692      sw      $v1,0($v0)

```

```

693 $L67:
694     lw      $v0,52($fp)
695     sw      $v0,28($fp)
696     sb      $zero,56($fp)
697     lw      $v1,28($fp)
698     li      $v0,-1                # 0xffffffffffffffff
699     bne     $v1,$v0,$L70
700     lbu     $v0,24($fp)
701     bne     $v0,$zero,$L70
702     b       $L69
703 $L70:
704     lw      $v1,28($fp)
705     li      $v0,10                # 0xa
706     bne     $v1,$v0,$L68
707     lbu     $v0,24($fp)
708     bne     $v0,$zero,$L69
709     b       $L68
710 $L69:
711     li      $v0,1                # 0x1
712     sb      $v0,56($fp)
713 $L68:
714     lbu     $v0,56($fp)
715     sb      $v0,36($fp)
716     lbu     $v0,36($fp)
717     bne     $v0,$zero,$L61
718     lw      $v0,__$F+4
719     addu    $v0,$v0,-1
720     sw      $v0,__$F+4
721     bgez    $v0,$L72
722     la      $a0,__$F
723     la      $t9,__$srget
724     jal     $ra,$t9
725     sw      $v0,60($fp)
726     b       $L73
727 $L72:
728     la      $v0,__$F
729     lw      $v1,0($v0)
730     move    $a0,$v1
731     lbu     $a0,0($a0)
732     sw      $a0,60($fp)
733     addu    $v1,$v1,1
734     sw      $v1,0($v0)
735 $L73:
736     lw      $v0,60($fp)
737     sw      $v0,32($fp)
738     b       $L61
739 $L53:
740     move    $sp,$fp
741     lw      $ra,72($sp)
742     lw      $fp,68($sp)
743     addu    $sp,$sp,80
744     j       $ra
745     .end    EntradaDecoderStandar
746     .size   EntradaDecoderStandar,.-EntradaDecoderStandar
747     .rdata
748     .align  2
749 $LC24:
750     .ascii  "encode\000"
751     .align  2
752 $LC25:
753     .ascii  "decode\000"

```

```

754     .text
755     .align 2
756     .globl comprobarAction
757     .ent comprobarAction
758 comprobarAction:
759     .frame $fp,40,$ra # vars= 0, regs= 3/0, args= 16, extra= 8
760     .mask 0xd0000000,-8
761     .fmask 0x00000000,0
762     .set noreorder
763     .cpload $t9
764     .set reorder
765     subu $sp,$sp,40
766     .cpstore 16
767     sw $ra,32($sp)
768     sw $fp,28($sp)
769     sw $gp,24($sp)
770     move $fp,$sp
771     sw $a0,40($fp)
772     lw $a0,40($fp)
773     la $a1,$LC24
774     la $t9,strcmp
775     jal $ra,$t9
776     bne $v0,$zero,$L75
777     li $v0,1 # 0x1
778     sb $v0,encoderActivo
779     sb $zero,decoderActivo
780 $L75:
781     lw $a0,40($fp)
782     la $a1,$LC25
783     la $t9,strcmp
784     jal $ra,$t9
785     bne $v0,$zero,$L74
786     sb $zero,encoderActivo
787     li $v0,1 # 0x1
788     sb $v0,decoderActivo
789 $L74:
790     move $sp,$fp
791     lw $ra,32($sp)
792     lw $fp,28($sp)
793     addu $sp,$sp,40
794     j $ra
795     .end comprobarAction
796     .size comprobarAction,.-comprobarAction
797     .rdata
798     .align 2
799 $LC26:
800     .ascii "Usage:\n\000"
801     .align 2
802 $LC27:
803     .ascii "\t ./tp0 -h\n\000"
804     .align 2
805 $LC28:
806     .ascii "\t ./tp0 -v\n\000"
807     .align 2
808 $LC29:
809     .ascii "Options:\n\000"
810     .align 2
811 $LC30:
812     .ascii "\t -v, --version, Shows the version of TP. \n\000"
813     .align 2
814 $LC31:

```

```

815     .ascii  "\t -h, --help , Show help \n\000"
816     .align  2
817 $LC32:
818     .ascii  "\t -i, --input, Location of the input file\n\000"
819     .align  2
820 $LC33:
821     .ascii  "\t -o, --output, Location of the output file\n\000"
822     .align  2
823 $LC34:
824     .ascii  "\t -a, --action, Program action: encode (default) or dec"
825     .ascii  "ode \n\000"
826     .align  2
827 $LC35:
828     .ascii  "Example: \n\000"
829     .align  2
830 $LC36:
831     .ascii  "\t ./tp0 -a encode -i /input -o /output -h\n\000"
832     .align  2
833 $LC37:
834     .ascii  "\t ./tp0 -a decode\n\000"
835     .text
836     .align  2
837     .globl  imprimirAyuda
838     .ent    imprimirAyuda
839 imprimirAyuda:
840     .frame  $fp,40,$ra          # vars= 0, regs= 3/0, args= 16, extra= 8
841     .mask   0xd0000000,-8
842     .fmask  0x00000000,0
843     .set    noreorder
844     .cpld   $t9
845     .set    reorder
846     subu    $sp,$sp,40
847     .cprestore 16
848     sw      $ra,32($sp)
849     sw      $fp,28($sp)
850     sw      $gp,24($sp)
851     move    $fp,$sp
852     la      $a0,$LC26
853     la      $t9,printf
854     jal     $ra,$t9
855     la      $a0,$LC27
856     la      $t9,printf
857     jal     $ra,$t9
858     la      $a0,$LC28
859     la      $t9,printf
860     jal     $ra,$t9
861     la      $a0,$LC29
862     la      $t9,printf
863     jal     $ra,$t9
864     la      $a0,$LC30
865     la      $t9,printf
866     jal     $ra,$t9
867     la      $a0,$LC31
868     la      $t9,printf
869     jal     $ra,$t9
870     la      $a0,$LC32
871     la      $t9,printf
872     jal     $ra,$t9
873     la      $a0,$LC33
874     la      $t9,printf
875     jal     $ra,$t9

```

```

876     la      $a0,$LC34
877     la      $t9,printf
878     jal     $ra,$t9
879     la      $a0,$LC35
880     la      $t9,printf
881     jal     $ra,$t9
882     la      $a0,$LC36
883     la      $t9,printf
884     jal     $ra,$t9
885     la      $a0,$LC37
886     la      $t9,printf
887     jal     $ra,$t9
888     move    $sp,$fp
889     lw      $ra,32($sp)
890     lw      $fp,28($sp)
891     addu    $sp,$sp,40
892     j       $ra
893     .end    imprimirAyuda
894     .size   imprimirAyuda, .-imprimirAyuda
895     .rdata
896     .align  2
897 $LC38:
898     .ascii  "vhi:o:a:\000"
899     .align  2
900 $LC39:
901     .ascii  "66.20-Organizacion de Computadoras TP Version 0.0\n\000"
902     .align  2
903 $LC40:
904     .ascii  "r\000"
905     .align  2
906 $LC41:
907     .ascii  "Error al abrir el archivo input %s\n\000"
908     .align  2
909 $LC42:
910     .ascii  "w\000"
911     .align  2
912 $LC43:
913     .ascii  "Error al abrir el archivo output %s \n\000"
914     .text
915     .align  2
916     .globl  opciones
917     .ent    opciones
918 opciones:
919     .frame  $fp,64,$ra          # vars= 16, regs= 3/0, args= 24, extra= 8
920     .mask   0xd0000000,-8
921     .fmask  0x00000000,0
922     .set    noreorder
923     .cpld   $t9
924     .set    reorder
925     subu    $sp,$sp,64
926     .cprestore 24
927     sw      $ra,56($sp)
928     sw      $fp,52($sp)
929     sw      $gp,48($sp)
930     move    $fp,$sp
931     sw      $a0,64($fp)
932     sw      $a1,68($fp)
933     sw      $zero,32($fp)
934     addu    $v0,$fp,32
935     sw      $v0,16($sp)
936     lw      $a0,64($fp)

```

```

937     lw      $a1,68($fp)
938     la      $a2,$LC38
939     la      $a3,long_options
940     la      $t9,getopt_long
941     jal     $ra,$t9
942     sw      $v0,36($fp)
943 $L79:
944     lw      $v1,36($fp)
945     li      $v0,-1                # 0xffffffffffffffff
946     bne     $v1,$v0,$L81
947     b       $L80
948 $L81:
949     lw      $v0,36($fp)
950     addu    $v0,$v0,-97
951     sw      $v0,44($fp)
952     lw      $v1,44($fp)
953     sltu    $v0,$v1,22
954     beq     $v0,$zero,$L82
955     lw      $v0,44($fp)
956     sll     $v1,$v0,2
957     la      $v0,$L91
958     addu    $v0,$v1,$v0
959     lw      $v0,0($v0)
960     .cpadd  $v0
961     j       $v0
962     .rdata
963     .align  2
964 $L91:
965     .gpword $L89
966     .gpword $L82
967     .gpword $L82
968     .gpword $L82
969     .gpword $L82
970     .gpword $L82
971     .gpword $L82
972     .gpword $L84
973     .gpword $L85
974     .gpword $L82
975     .gpword $L82
976     .gpword $L82
977     .gpword $L82
978     .gpword $L82
979     .gpword $L87
980     .gpword $L82
981     .gpword $L82
982     .gpword $L82
983     .gpword $L82
984     .gpword $L82
985     .gpword $L82
986     .gpword $L83
987     .text
988 $L83:
989     la      $a0,$LC39
990     la      $t9,printf
991     jal     $ra,$t9
992     li      $v0,1                # 0x1
993     sw      $v0,40($fp)
994     b       $L78
995 $L84:
996     la      $t9,imprimirAyuda
997     jal     $ra,$t9

```

```

998      li      $v1,1                      # 0x1
999      sw      $v1,40($fp)
1000     b       $L78
1001     $L85:
1002     lw      $a0,optarg
1003     la      $a1,$LC40
1004     la      $t9,fopen
1005     jal     $ra,$t9
1006     sw      $v0,finput
1007     lw      $v0,finput
1008     bne     $v0,$zero,$L82
1009     la      $a0,__$SF+176
1010     la      $a1,$LC41
1011     lw      $a2,optarg
1012     la      $t9,fprintf
1013     jal     $ra,$t9
1014     li      $a0,1                      # 0x1
1015     la      $t9,exit
1016     jal     $ra,$t9
1017     $L87:
1018     lw      $a0,optarg
1019     la      $a1,$LC42
1020     la      $t9,fopen
1021     jal     $ra,$t9
1022     sw      $v0,foutput
1023     lw      $v0,foutput
1024     bne     $v0,$zero,$L82
1025     la      $a0,__$SF+176
1026     la      $a1,$LC43
1027     lw      $a2,optarg
1028     la      $t9,fprintf
1029     jal     $ra,$t9
1030     li      $a0,1                      # 0x1
1031     la      $t9,exit
1032     jal     $ra,$t9
1033     $L89:
1034     lw      $a0,optarg
1035     la      $t9,comprobarAction
1036     jal     $ra,$t9
1037     $L82:
1038     addu    $v0,$fp,32
1039     sw      $v0,16($sp)
1040     lw      $a0,64($fp)
1041     lw      $a1,68($fp)
1042     la      $a2,$LC38
1043     la      $a3,long_options
1044     la      $t9,getopt_long
1045     jal     $ra,$t9
1046     sw      $v0,36($fp)
1047     b       $L79
1048     $L80:
1049     sw      $zero,40($fp)
1050     $L78:
1051     lw      $v0,40($fp)
1052     move    $sp,$fp
1053     lw      $ra,56($sp)
1054     lw      $fp,52($sp)
1055     addu    $sp,$sp,64
1056     j       $ra
1057     .end    opciones
1058     .size   opciones,.-opciones

```



```

1059     .align    2
1060     .globl    controlarOpciones
1061     .ent      controlarOpciones
1062 controlarOpciones:
1063     .frame    $fp,40,$ra                # vars= 0, regs= 3/0, args= 16, extra= 8
1064     .mask     0xd0000000,-8
1065     .fmask    0x00000000,0
1066     .set      noreorder
1067     .cpload   $t9
1068     .set      reorder
1069     subu      $sp,$sp,40
1070     .cprestore 16
1071     sw        $ra,32($sp)
1072     sw        $fp,28($sp)
1073     sw        $gp,24($sp)
1074     move      $fp,$sp
1075     lw        $v0,finput
1076     beq       $v0,$zero,$L93
1077     lw        $a0,finput
1078     lw        $a1,foutput
1079     la        $t9,procesarArchivos
1080     jal       $ra,$t9
1081     b         $L92
1082 $L93:
1083     lbu       $v0,encoderActivo
1084     beq       $v0,$zero,$L95
1085     la        $t9,EntradaEncoderStandar
1086     jal       $ra,$t9
1087     b         $L92
1088 $L95:
1089     la        $t9,EntradaDecoderStandar
1090     jal       $ra,$t9
1091 $L92:
1092     move      $sp,$fp
1093     lw        $ra,32($sp)
1094     lw        $fp,28($sp)
1095     addu      $sp,$sp,40
1096     j         $ra
1097     .end      controlarOpciones
1098     .size     controlarOpciones,.-controlarOpciones
1099     .align    2
1100     .globl    main
1101     .ent      main
1102 main:
1103     .frame    $fp,48,$ra                # vars= 8, regs= 3/0, args= 16, extra= 8
1104     .mask     0xd0000000,-8
1105     .fmask    0x00000000,0
1106     .set      noreorder
1107     .cpload   $t9
1108     .set      reorder
1109     subu      $sp,$sp,48
1110     .cprestore 16
1111     sw        $ra,40($sp)
1112     sw        $fp,36($sp)
1113     sw        $gp,32($sp)
1114     move      $fp,$sp
1115     sw        $a0,48($fp)
1116     sw        $a1,52($fp)
1117     lw        $a0,48($fp)
1118     lw        $a1,52($fp)
1119     la        $t9,opciones

```

```

1120      jal      $ra,$t9
1121      sw       $v0,24($fp)
1122      lw       $v0,24($fp)
1123      bne      $v0,$zero,$L98
1124      la       $t9,controlarOpciones
1125      jal      $ra,$t9
1126 $L98:
1127      move     $v0,$zero
1128      move     $sp,$fp
1129      lw       $ra,40($sp)
1130      lw       $fp,36($sp)
1131      addu     $sp,$sp,48
1132      j        $ra
1133      .end     main
1134      .size    main,.-main
1135      .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 huésped.