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. Código Fuente

2.1. Código fuente C

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



bool encoderActivo = true;
bool decoderActivo = false;

FILE* finput = NULL;
FILE* foutput = NULL;



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}
};


char* encoder( int numInt){

        int highNible = numInt & 0xf0;
        int lowNible = numInt & 0x0f;
        int primerNum = highNible >> 4;
        int segundoNum = lowNible;

        char* primerChar = vecHexa[primerNum];
        char* segundoChar = vecHexa[segundoNum];
        char* valorHexa = malloc( sizeof(char)*3 );

        strcpy(valorHexa,primerChar);
        strcat(valorHexa,segundoChar);

        return valorHexa;
}


int correrReferencia ( int numInt ){
        if( numInt > 47 && numInt < 58)
                return numInt - 48 ;

        else if( numInt > 64 && numInt < 71)
                return numInt - 55 ;

```

```
55          else if( numInt > 96 && numInt < 103)
56                  return numInt - 87 ;
57          else {
58                  fprintf(stderr,"Contiene caracteres que no pertenecen al codigo Hexa\
                        n");
59                  exit(1);
60          }
61
62  }
63
64  char decoder ( int numPri, int numSeg){
65
66          int valor1 = correrReferencia( numPri );
67          int valor2 = correrReferencia( numSeg );
68
69          int highNible = valor1 << 4;
70          highNible =  highNible & 0xf0;
71          valor2 = valor2 & 0x0f;
72
73          char caracter = highNible | valor2;
74
75          return caracter;
76
77
78  }
79
80  void procesarArchivos(FILE* finput, FILE* foutput){
81
82          char* string;
83          char c;
84          int caracter2;
85          int caracter = fgetc(finput);
86
87          while (caracter != EOF){
88                  if(encoderActivo){
89                          string = encoder(caracter);
90                          if(foutput != NULL) fputs(string , foutput);
91                          else printf("%s",string);
92                          free(string);
93                  }
94                  else{
95
96                          caracter2 = fgetc(finput);
97                          c = decoder(caracter,caracter2);
98                          if(foutput != NULL) fputc(c,foutput);
99                          else printf("%c",c);
100
101
102                 }
103                 caracter = fgetc(finput);
104         }
105
106     fclose(finput);
107     if(foutput != NULL) fclose(foutput);
108 }
109
110 void EntradaEncoderStandar(){
111
112         int c = getchar();
113         while( c != EOF && c!='\n'){
114                 char* string = encoder(c);
```

```
115                 printf("%s",string);
116                 free(string);
117                 c = getchar();
118         }
119 }
120
121 void EntradaDecoderStandar(){
122         int c = getchar();
123         int c2 = getchar();
124         while( c != EOF && c!='\n' ){
125                 char string = decoder(c,c2);
126                 if( foutput != NULL ) fputc(string,foutput);
127                 else printf("%c",string);
128                 c = getchar();
129                 c2 = getchar();
130         }
131
132 }
133
134
135 void comprobarAction(char* optarg){
136         if( strcmp ( optarg, "encode" ) == 0 ){
137                 encoderActivo = true;
138                 decoderActivo = false;
139         }
140         if ( strcmp ( optarg, "decode" ) == 0 ){
141                 encoderActivo = false;
142                 decoderActivo = true;
143         }
144
145 }
146
147 void imprimirAyuda(){
148         printf("Usage:\n");
149         printf("\t ./tp0 -h\n");
150         printf("\t ./tp0 -v\n");
151         printf("Options:\n");
152         printf("\t -v, --version, Shows the version of TP. \n");
153         printf("\t -h, --help , Show help \n");
154         printf("\t -i, --input, Location of the input file\n");
155         printf("\t -o, --output, Location of the output file\n");
156         printf("\t -a, --action, Program action: encode (default) or decode \n");
157         printf("Example: \n");
158         printf("\t ./tp0 -a encode -i /input -o /output -h\n");
159         printf("\t ./tp0 -a decode\n");
160 }
161 int opciones( int argc , char** argv ){
162
163         int option_index = 0;
164         int option = getopt_long ( argc, argv, "vhi:o:a:", long_options, &
              option_index);
165         while ( option != -1 ){
166
167                 switch ( option ){
168                         case 'v':
169                                         printf("66.20-Organizacion de Computadoras TP
                                                Version 0.0\n");
170                                         return 1;
171                                         break;
172                         case 'h':
173                                         imprimirAyuda();
```

```
                                        return 1;
                                        break;

                        case 'i':
                                        finput = fopen(optarg,"r");
                                        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;
}


void controlarOpciones(){

         if (finput != NULL){
                procesarArchivos(finput,foutput);
        }
        else{
                if(encoderActivo) EntradaEncoderStandar();
                else EntradaDecoderStandar();
        }

}

int main (int argc, char** argv){
        int opcion = opciones( argc , argv);

        if(opcion == 0){
                controlarOpciones();
                printf("\nSe completo con exito la operacion\n");
        }
        return 0;

}
```

2.2. Código assembly MIPS

```
        .file   1 "tp0.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .globl  encoderActivo
        .data
        .type   encoderActivo, @object
        .size   encoderActivo, 1
encoderActivo:
        .byte   1
        .globl  decoderActivo
        .globl  decoderActivo
        .section        .bss
        .align  0
        .type   decoderActivo, @object
        .size   decoderActivo, 1
decoderActivo:
        .space  1
        .globl  finput
        .globl  finput
        .align  2
        .type   finput, @object
        .size   finput, 4
finput:
        .space  4
        .globl  foutput
        .globl  foutput
        .align  2
        .type   foutput, @object
        .size   foutput, 4
foutput:
        .space  4
        .rdata
        .align  2
$LC0:
        .ascii  "0\000"
        .align  2
$LC1:
        .ascii  "1\000"
        .align  2
$LC2:
        .ascii  "2\000"
        .align  2
$LC3:
        .ascii  "3\000"
        .align  2
$LC4:
        .ascii  "4\000"
        .align  2
$LC5:
        .ascii  "5\000"
        .align  2
$LC6:
        .ascii  "6\000"
        .align  2
$LC7:
        .ascii  "7\000"
        .align  2
```

```
$LC8:
        .ascii  "8\000"
        .align  2
$LC9:
        .ascii  "9\000"
        .align  2
$LC10:
        .ascii  "A\000"
        .align  2
$LC11:
        .ascii  "B\000"
        .align  2
$LC12:
        .ascii  "C\000"
        .align  2
$LC13:
        .ascii  "D\000"
        .align  2
$LC14:
        .ascii  "E\000"
        .align  2
$LC15:
        .ascii  "F\000"
        .globl  vecHexa
        .data
        .align  2
        .type   vecHexa, @object
        .size   vecHexa, 64
vecHexa:
        .word   $LC0
        .word   $LC1
        .word   $LC2
        .word   $LC3
        .word   $LC4
        .word   $LC5
        .word   $LC6
        .word   $LC7
        .word   $LC8
        .word   $LC9
        .word   $LC10
        .word   $LC11
        .word   $LC12
        .word   $LC13
        .word   $LC14
        .word   $LC15
        .rdata
        .align  2
$LC16:
        .ascii  "version\000"
        .align  2
$LC17:
        .ascii  "help\000"
        .align  2
$LC18:
        .ascii  "input\000"
        .align  2
$LC19:
        .ascii  "output\000"
        .align  2
$LC20:
        .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
```

```
$L19:
        lw      $v0,48($fp)
        slt     $v0,$v0,65
        bne     $v0,$zero,$L21
        lw      $v0,48($fp)
        slt     $v0,$v0,71
        beq     $v0,$zero,$L21
        lw      $v0,48($fp)
        addu    $v0,$v0,-55
        sw      $v0,24($fp)
        b       $L18
$L21:
        lw      $v0,48($fp)
        slt     $v0,$v0,97
        bne     $v0,$zero,$L23
        lw      $v0,48($fp)
        slt     $v0,$v0,103
        beq     $v0,$zero,$L23
        lw      $v0,48($fp)
        addu    $v0,$v0,-87
        sw      $v0,24($fp)
        b       $L18
$L23:
        la      $a0,__sF+176
        la      $a1,$LC21
        la      $t9,fprintf
        jal     $ra,$t9
        li      $a0,1                   # 0x1
        la      $t9,exit
        jal     $ra,$t9
$L18:
        lw      $v0,24($fp)
        move    $sp,$fp
        lw      $ra,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        j       $ra
        .end    correrReferencia
        .size   correrReferencia, .-correrReferencia
        .align  2
        .globl  decoder
        .ent    decoder
decoder:
        .frame  $fp,56,$ra              # vars= 16, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,56
        .cprestore 16
        sw      $ra,48($sp)
        sw      $fp,44($sp)
        sw      $gp,40($sp)
        move    $fp,$sp
        sw      $a0,56($fp)
        sw      $a1,60($fp)
        lw      $a0,56($fp)
        la      $t9,correrReferencia
        jal     $ra,$t9
        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         .cpload $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,__sF+4
456         addu    $v0,$v0,-1
457         sw      $v0,__sF+4
458         bgez    $v0,$L38
459         la      $a0,__sF
460         la      $t9,__srget
461         jal     $ra,$t9
462         sw      $v0,40($fp)
463         b       $L39
464 $L38:
465         la      $v0,__sF
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
```

```
        lbu     $v0,24($fp)
        bne     $v0,$zero,$L41
        b       $L40
$L41:
        li      $v0,1                   # 0x1
        sb      $v0,44($fp)
$L40:
        lbu     $v0,44($fp)
        sb      $v0,32($fp)
$L43:
        lbu     $v0,32($fp)
        beq     $v0,$zero,$L45
        b       $L37
$L45:
        lw      $a0,28($fp)
        la      $t9,encoder
        jal     $ra,$t9
        sw      $v0,36($fp)
        la      $a0,$LC22
        lw      $a1,36($fp)
        la      $t9,printf
        jal     $ra,$t9
        lw      $a0,36($fp)
        la      $t9,free
        jal     $ra,$t9
        lw      $v0,__sF+4
        addu    $v0,$v0,-1
        sw      $v0,__sF+4
        bgez    $v0,$L46
        la      $a0,__sF
        la      $t9,__srget
        jal     $ra,$t9
        sw      $v0,48($fp)
        b       $L47
$L46:
        la      $v0,__sF
        lw      $v1,0($v0)
        move    $a0,$v1
        lbu     $a0,0($a0)
        sw      $a0,48($fp)
        addu    $v1,$v1,1
        sw      $v1,0($v0)
$L47:
        lw      $v0,48($fp)
        sw      $v0,28($fp)
        sb      $zero,52($fp)
        lw      $v1,28($fp)
        li      $v0,-1                  # 0xffffffffffffffff
        bne     $v1,$v0,$L50
        lbu     $v0,24($fp)
        bne     $v0,$zero,$L50
        b       $L49
$L50:
        lw      $v1,28($fp)
        li      $v0,10                  # 0xa
        bne     $v1,$v0,$L48
        lbu     $v0,24($fp)
        bne     $v0,$zero,$L49
        b       $L48
$L49:
        li      $v0,1                   # 0x1
```

```
547           sb      $v0,52($fp)
548   $L48:
549           lbu     $v0,52($fp)
550           sb      $v0,32($fp)
551           b       $L43
552   $L37:
553           move    $sp,$fp
554           lw      $ra,64($sp)
555           lw      $fp,60($sp)
556           addu    $sp,$sp,72
557           j       $ra
558           .end    EntradaEncoderStandar
559           .size   EntradaEncoderStandar, .-EntradaEncoderStandar
560           .align  2
561           .globl  EntradaDecoderStandar
562           .ent    EntradaDecoderStandar
563   EntradaDecoderStandar:
564           .frame  $fp,80,$ra              # vars= 40, regs= 3/0, args= 16, extra= 8
565           .mask   0xd0000000,-8
566           .fmask  0x00000000,0
567           .set    noreorder
568           .cpload $t9
569           .set    reorder
570           subu    $sp,$sp,80
571           .cprestore 16
572           sw      $ra,72($sp)
573           sw      $fp,68($sp)
574           sw      $gp,64($sp)
575           move    $fp,$sp
576           move    $a0,$zero
577           la      $t9,isatty
578           jal     $ra,$t9
579           sltu    $v0,$zero,$v0
580           sb      $v0,24($fp)
581           lw      $v0,__sF+4
582           addu    $v0,$v0,-1
583           sw      $v0,__sF+4
584           bgez    $v0,$L52
585           la      $a0,__sF
586           la      $t9,__srget
587           jal     $ra,$t9
588           sw      $v0,40($fp)
589           b       $L53
590   $L52:
591           la      $v0,__sF
592           lw      $v1,0($v0)
593           move    $a0,$v1
594           lbu     $a0,0($a0)
595           sw      $a0,40($fp)
596           addu    $v1,$v1,1
597           sw      $v1,0($v0)
598   $L53:
599           lw      $v0,40($fp)
600           sw      $v0,28($fp)
601           lw      $v0,__sF+4
602           addu    $v0,$v0,-1
603           sw      $v0,__sF+4
604           bgez    $v0,$L54
605           la      $a0,__sF
606           la      $t9,__srget
607           jal     $ra,$t9
```

```
608           sw      $v0,44($fp)
609           b       $L55
610   $L54:
611           la      $v0,__sF
612           lw      $v1,0($v0)
613           move    $a0,$v1
614           lbu     $a0,0($a0)
615           sw      $a0,44($fp)
616           addu    $v1,$v1,1
617           sw      $v1,0($v0)
618   $L55:
619           lw      $v0,44($fp)
620           sw      $v0,32($fp)
621           sb      $zero,48($fp)
622           lw      $v1,28($fp)
623           li      $v0,-1                  # 0xffffffffffffffff
624           bne     $v1,$v0,$L58
625           lbu     $v0,24($fp)
626           bne     $v0,$zero,$L58
627           b       $L57
628   $L58:
629           lw      $v1,28($fp)
630           li      $v0,10                  # 0xa
631           bne     $v1,$v0,$L56
632           lbu     $v0,24($fp)
633           bne     $v0,$zero,$L57
634           b       $L56
635   $L57:
636           li      $v0,1                   # 0x1
637           sb      $v0,48($fp)
638   $L56:
639           lbu     $v0,48($fp)
640           sb      $v0,36($fp)
641   $L59:
642           lbu     $v0,36($fp)
643           beq     $v0,$zero,$L61
644           b       $L51
645   $L61:
646           lw      $a0,28($fp)
647           lw      $a1,32($fp)
648           la      $t9,decoder
649           jal     $ra,$t9
650           sb      $v0,37($fp)
651           lw      $v0,foutput
652           beq     $v0,$zero,$L62
653           lb      $v0,37($fp)
654           move    $a0,$v0
655           lw      $a1,foutput
656           la      $t9,fputc
657           jal     $ra,$t9
658           b       $L63
659   $L62:
660           lb      $v0,37($fp)
661           la      $a0,$LC23
662           move    $a1,$v0
663           la      $t9,printf
664           jal     $ra,$t9
665   $L63:
666           lw      $v0,__sF+4
667           addu    $v0,$v0,-1
668           sw      $v0,__sF+4
```

```
669           bgez    $v0,$L64
670           la      $a0,__sF
671           la      $t9,__srget
672           jal     $ra,$t9
673           sw      $v0,52($fp)
674           b       $L65
675   $L64:
676           la      $v0,__sF
677           lw      $v1,0($v0)
678           move    $a0,$v1
679           lbu     $a0,0($a0)
680           sw      $a0,52($fp)
681           addu    $v1,$v1,1
682           sw      $v1,0($v0)
683   $L65:
684           lw      $v0,52($fp)
685           sw      $v0,28($fp)
686           lw      $v0,__sF+4
687           addu    $v0,$v0,-1
688           sw      $v0,__sF+4
689           bgez    $v0,$L66
690           la      $a0,__sF
691           la      $t9,__srget
692           jal     $ra,$t9
693           sw      $v0,56($fp)
694           b       $L67
695   $L66:
696           la      $v0,__sF
697           lw      $v1,0($v0)
698           move    $a0,$v1
699           lbu     $a0,0($a0)
700           sw      $a0,56($fp)
701           addu    $v1,$v1,1
702           sw      $v1,0($v0)
703   $L67:
704           lw      $v0,56($fp)
705           sw      $v0,32($fp)
706           sb      $zero,60($fp)
707           lw      $v1,28($fp)
708           li      $v0,-1                  # 0xffffffffffffffff
709           bne     $v1,$v0,$L70
710           lbu     $v0,24($fp)
711           bne     $v0,$zero,$L70
712           b       $L69
713   $L70:
714           lw      $v1,28($fp)
715           li      $v0,10                  # 0xa
716           bne     $v1,$v0,$L68
717           lbu     $v0,24($fp)
718           bne     $v0,$zero,$L69
719           b       $L68
720   $L69:
721           li      $v0,1                   # 0x1
722           sb      $v0,60($fp)
723   $L68:
724           lbu     $v0,60($fp)
725           sb      $v0,36($fp)
726           b       $L59
727   $L51:
728           move    $sp,$fp
729           lw      $ra,72($sp)
```

```
730         lw      $fp,68($sp)
731         addu    $sp,$sp,80
732         j       $ra
733         .end    EntradaDecoderStandar
734         .size   EntradaDecoderStandar, .-EntradaDecoderStandar
735         .rdata
736         .align  2
737 $LC24:
738         .ascii  "encode\000"
739         .align  2
740 $LC25:
741         .ascii  "decode\000"
742         .text
743         .align  2
744         .globl  comprobarAction
745         .ent    comprobarAction
746 comprobarAction:
747         .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16, extra= 8
748         .mask   0xd0000000,-8
749         .fmask  0x00000000,0
750         .set    noreorder
751         .cpload $t9
752         .set    reorder
753         subu    $sp,$sp,40
754         .cprestore 16
755         sw      $ra,32($sp)
756         sw      $fp,28($sp)
757         sw      $gp,24($sp)
758         move    $fp,$sp
759         sw      $a0,40($fp)
760         lw      $a0,40($fp)
761         la      $a1,$LC24
762         la      $t9,strcmp
763         jal     $ra,$t9
764         bne     $v0,$zero,$L72
765         li      $v0,1                   # 0x1
766         sb      $v0,encoderActivo
767         sb      $zero,decoderActivo
768 $L72:
769         lw      $a0,40($fp)
770         la      $a1,$LC25
771         la      $t9,strcmp
772         jal     $ra,$t9
773         bne     $v0,$zero,$L71
774         sb      $zero,encoderActivo
775         li      $v0,1                   # 0x1
776         sb      $v0,decoderActivo
777 $L71:
778         move    $sp,$fp
779         lw      $ra,32($sp)
780         lw      $fp,28($sp)
781         addu    $sp,$sp,40
782         j       $ra
783         .end    comprobarAction
784         .size   comprobarAction, .-comprobarAction
785         .rdata
786         .align  2
787 $LC26:
788         .ascii  "Usage:\n\000"
789         .align  2
790 $LC27:
```

```
791           .ascii  "\t ./tp0 -h\n\000"
792           .align  2
793   $LC28:
794           .ascii  "\t ./tp0 -v\n\000"
795           .align  2
796   $LC29:
797           .ascii  "Options:\n\000"
798           .align  2
799   $LC30:
800           .ascii  "\t -v, --version, Shows the version of TP. \n\000"
801           .align  2
802   $LC31:
803           .ascii  "\t -h, --help , Show help \n\000"
804           .align  2
805   $LC32:
806           .ascii  "\t -i, --input, Location of the input file\n\000"
807           .align  2
808   $LC33:
809           .ascii  "\t -o, --output, Location of the output file\n\000"
810           .align  2
811   $LC34:
812           .ascii  "\t -a, --action, Program action: encode (default) or dec"
813           .ascii  "ode \n\000"
814           .align  2
815   $LC35:
816           .ascii  "Example: \n\000"
817           .align  2
818   $LC36:
819           .ascii  "\t ./tp0 -a encode -i /input -o /output -h\n\000"
820           .align  2
821   $LC37:
822           .ascii  "\t ./tp0 -a decode\n\000"
823           .text
824           .align  2
825           .globl  imprimirAyuda
826           .ent    imprimirAyuda
827   imprimirAyuda:
828           .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16, extra= 8
829           .mask   0xd0000000,-8
830           .fmask  0x00000000,0
831           .set    noreorder
832           .cpload $t9
833           .set    reorder
834           subu    $sp,$sp,40
835           .cprestore 16
836           sw      $ra,32($sp)
837           sw      $fp,28($sp)
838           sw      $gp,24($sp)
839           move    $fp,$sp
840           la      $a0,$LC26
841           la      $t9,printf
842           jal     $ra,$t9
843           la      $a0,$LC27
844           la      $t9,printf
845           jal     $ra,$t9
846           la      $a0,$LC28
847           la      $t9,printf
848           jal     $ra,$t9
849           la      $a0,$LC29
850           la      $t9,printf
851           jal     $ra,$t9
```

```
852             la      $a0,$LC30
853             la      $t9,printf
854             jal     $ra,$t9
855             la      $a0,$LC31
856             la      $t9,printf
857             jal     $ra,$t9
858             la      $a0,$LC32
859             la      $t9,printf
860             jal     $ra,$t9
861             la      $a0,$LC33
862             la      $t9,printf
863             jal     $ra,$t9
864             la      $a0,$LC34
865             la      $t9,printf
866             jal     $ra,$t9
867             la      $a0,$LC35
868             la      $t9,printf
869             jal     $ra,$t9
870             la      $a0,$LC36
871             la      $t9,printf
872             jal     $ra,$t9
873             la      $a0,$LC37
874             la      $t9,printf
875             jal     $ra,$t9
876             move    $sp,$fp
877             lw      $ra,32($sp)
878             lw      $fp,28($sp)
879             addu    $sp,$sp,40
880             j       $ra
881             .end    imprimirAyuda
882             .size   imprimirAyuda, .-imprimirAyuda
883             .rdata
884             .align  2
885     $LC38:
886             .ascii  "vhi:o:a:\000"
887             .align  2
888     $LC39:
889             .ascii  "66.20-Organizacion de Computadoras TP Version 0.0\n\000"
890             .align  2
891     $LC40:
892             .ascii  "r\000"
893             .align  2
894     $LC41:
895             .ascii  "Error al abrir el archivo input %s\n\000"
896             .align  2
897     $LC42:
898             .ascii  "w\000"
899             .align  2
900     $LC43:
901             .ascii  "Error al abrir el archivo output %s \n\000"
902             .text
903             .align  2
904             .globl  opciones
905             .ent    opciones
906     opciones:
907             .frame  $fp,64,$ra              # vars= 16, regs= 3/0, args= 24, extra= 8
908             .mask   0xd0000000,-8
909             .fmask  0x00000000,0
910             .set    noreorder
911             .cpload $t9
912             .set    reorder
```

```
913           subu    $sp,$sp,64
914           .cprestore 24
915           sw      $ra,56($sp)
916           sw      $fp,52($sp)
917           sw      $gp,48($sp)
918           move    $fp,$sp
919           sw      $a0,64($fp)
920           sw      $a1,68($fp)
921           sw      $zero,32($fp)
922           addu    $v0,$fp,32
923           sw      $v0,16($sp)
924           lw      $a0,64($fp)
925           lw      $a1,68($fp)
926           la      $a2,$LC38
927           la      $a3,long_options
928           la      $t9,getopt_long
929           jal     $ra,$t9
930           sw      $v0,36($fp)
931   $L76:
932           lw      $v1,36($fp)
933           li      $v0,-1                  # 0xffffffffffffffff
934           bne     $v1,$v0,$L78
935           b       $L77
936   $L78:
937           lw      $v0,36($fp)
938           addu    $v0,$v0,-97
939           sw      $v0,44($fp)
940           lw      $v1,44($fp)
941           sltu    $v0,$v1,22
942           beq     $v0,$zero,$L79
943           lw      $v0,44($fp)
944           sll     $v1,$v0,2
945           la      $v0,$L88
946           addu    $v0,$v1,$v0
947           lw      $v0,0($v0)
948           .cpadd  $v0
949           j       $v0
950           .rdata
951           .align  2
952   $L88:
953           .gpword $L86
954           .gpword $L79
955           .gpword $L79
956           .gpword $L79
957           .gpword $L79
958           .gpword $L79
959           .gpword $L79
960           .gpword $L81
961           .gpword $L82
962           .gpword $L79
963           .gpword $L79
964           .gpword $L79
965           .gpword $L79
966           .gpword $L79
967           .gpword $L84
968           .gpword $L79
969           .gpword $L79
970           .gpword $L79
971           .gpword $L79
972           .gpword $L79
973           .gpword $L79
```

```
974           .gpword $L80
975           .text
976   $L80:
977           la      $a0,$LC39
978           la      $t9,printf
979           jal     $ra,$t9
980           li      $v0,1                   # 0x1
981           sw      $v0,40($fp)
982           b       $L75
983   $L81:
984           la      $t9,imprimirAyuda
985           jal     $ra,$t9
986           li      $v1,1                   # 0x1
987           sw      $v1,40($fp)
988           b       $L75
989   $L82:
990           lw      $a0,optarg
991           la      $a1,$LC40
992           la      $t9,fopen
993           jal     $ra,$t9
994           sw      $v0,finput
995           lw      $v0,finput
996           bne     $v0,$zero,$L79
997           la      $a0,__sF+176
998           la      $a1,$LC41
999           lw      $a2,optarg
1000          la      $t9,fprintf
1001          jal     $ra,$t9
1002          li      $a0,1                   # 0x1
1003          la      $t9,exit
1004          jal     $ra,$t9
1005  $L84:
1006          lw      $a0,optarg
1007          la      $a1,$LC42
1008          la      $t9,fopen
1009          jal     $ra,$t9
1010          sw      $v0,foutput
1011          lw      $v0,foutput
1012          bne     $v0,$zero,$L79
1013          la      $a0,__sF+176
1014          la      $a1,$LC43
1015          lw      $a2,optarg
1016          la      $t9,fprintf
1017          jal     $ra,$t9
1018          li      $a0,1                   # 0x1
1019          la      $t9,exit
1020          jal     $ra,$t9
1021  $L86:
1022          lw      $a0,optarg
1023          la      $t9,comprobarAction
1024          jal     $ra,$t9
1025  $L79:
1026          addu    $v0,$fp,32
1027          sw      $v0,16($sp)
1028          lw      $a0,64($fp)
1029          lw      $a1,68($fp)
1030          la      $a2,$LC38
1031          la      $a3,long_options
1032          la      $t9,getopt_long
1033          jal     $ra,$t9
1034          sw      $v0,36($fp)
```

```
        b       $L76
$L77:
        sw      $zero,40($fp)
$L75:
        lw      $v0,40($fp)
        move    $sp,$fp
        lw      $ra,56($sp)
        lw      $fp,52($sp)
        addu    $sp,$sp,64
        j       $ra
        .end    opciones
        .size   opciones, .-opciones
        .align  2
        .globl  controlarOpciones
        .ent    controlarOpciones
controlarOpciones:
        .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
        lw      $v0,finput
        beq     $v0,$zero,$L90
        lw      $a0,finput
        lw      $a1,foutput
        la      $t9,procesarArchivos
        jal     $ra,$t9
        b       $L89
$L90:
        lbu     $v0,encoderActivo
        beq     $v0,$zero,$L92
        la      $t9,EntradaEncoderStandar
        jal     $ra,$t9
        b       $L89
$L92:
        la      $t9,EntradaDecoderStandar
        jal     $ra,$t9
$L89:
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    controlarOpciones
        .size   controlarOpciones, .-controlarOpciones
        .align  2
        .globl  main
        .ent    main
main:
        .frame  $fp,48,$ra              # vars= 8, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
```

```
1096            .set    reorder
1097            subu    $sp,$sp,48
1098            .cprestore 16
1099            sw      $ra,40($sp)
1100            sw      $fp,36($sp)
1101            sw      $gp,32($sp)
1102            move    $fp,$sp
1103            sw      $a0,48($fp)
1104            sw      $a1,52($fp)
1105            lw      $a0,48($fp)
1106            lw      $a1,52($fp)
1107            la      $t9,opciones
1108            jal     $ra,$t9
1109            sw      $v0,24($fp)
1110            lw      $v0,24($fp)
1111            bne     $v0,$zero,$L95
1112            la      $t9,controlarOpciones
1113            jal     $ra,$t9
1114    $L95:
1115            move    $v0,$zero
1116            move    $sp,$fp
1117            lw      $ra,40($sp)
1118            lw      $fp,36($sp)
1119            addu    $sp,$sp,48
1120            j       $ra
1121            .end    main
1122            .size   main, .-main
1123            .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

3. Conclusiones