



# UBA FACULTAD DE INGENIERÍA

66.20 Organización de Computadoras

# Trabajo Práctico 0

$2^{do}$ Cuatrimestre 2017

**Integrantes:**

| | |
|---|---|
| Rodriquez Longhi, Federico | 93336 |
| federico.rlonghi@gmail.com | |
| Deciancio, Nicolás Andrés | 92150 |
| nicodec_89@hotmail.com | |
| Marshall, Juan Patricio | 95471 |
| juan.pmarshall@gmail.com | |

# Índice

# 1. Introducción

El trabajo práctico consistió en la elaboración de un programa escrito en lenguaje C, el cual consistía en el procesamiento de texto para determinar palabras palíndromas dentro del mismo. El código fue ejecutado sobre el sistema operativo linux y netbsd (provisto por el curso). Dentro del ambiente virtual se compilo el código para obtener la salida en código MIPS.

# 2. Documentación

El uso del programa se compone de las siguientes opciones que le son pasadas por parámetro:

- `-h` o `--help`: muestra la ayuda.
- `-V` o `--version`: muestra la versión.
- `-i` o `--input`: recibe como parámetro un archivo de texto como entrada. En caso de que no usar esta opción, se toma como entrada la entrada estándar.
- `-o` o `--output`: recibe como parámetro un archivo de texto como salida. En caso de que no usar esta opción, se toma como salida la salida estándar.

# 3. Compilación

El programa puede ser compilado ubicándose en la carpeta que contiene el código fuente tp0.c y correr el siguiente comando:

```
gcc -Wall -o "tp0tp0_2.c"
```

También se provee de un script `compilar` el cual nos compilará el código automáticamente:

```
./compilar
```

# 4. Pruebas

Para las pruebas se proveen de dos scripts que las ejecutan. El primer script `test.sh` ejecuta los ejemplos del enunciado.

El segundo script `test_p.sh` ejecuta las pruebas propias. Este archivo esta diseñado para poder agregar pruebas de forma sencilla, simplemente se debe agregar una linea en el sector de pruebas de la siguiente manera:

```
make_test <nombre><entrada de texto><salida esperada>
```

Este script crea los archivos correspondientes en la carpeta tests (dentro del directorio sobre el cual se ejecuta). Los archivos creados son de la forma:

- `test-<nombre del test>_in`: archivo de entrada
- `test-<nombre del test>_out`: archivo de salida generado por el programa
- `test-<nombre del test>_expected`: archivo de salida esperado

### 4.1. Corridas de Prueba

A continuación se muestran las corridas de prueba generadas por el script:

## 5. Conclusión

Al realizar este trabajo, comprendimos todo lo que implica simular un sistema operativo y lograr un entorno de desarrollo estable para trabajar a nivel MIPS. La apertura de un túnel reverso parecía que iba a ser una dificultad, pero con la documentación dada en la práctica no lo fue. Ayudo que todos los integrantes del grupo teníamos alguna experiencia en el uso de sistemas operativos UNIX. Gracias a los flags de compilación vistos, pudimos obtener el código assembly de nuestro programa escrito en C. Nos resulto muy interesante que un programa no muy complejo y de pocas lineas, convertido a assembly ocupara tantas lineas. Es lógico, viendo que C es un lenguaje de un nivel mucho mas alto que assembly, y se abstrae de muchos elementos de la arquitectura de la computadora, que en assembly son más relevantes. Como conclusión grupal, llegamos a que este práctico nos sirvió como iniciación a todo un mundo de desarrollo a bajo nivel, usando codigo MIPS, que previamente pasabamos bastante por alto.

```
1
2  Compiling Source
3  Compilation Success
4  Starting Tests
5
6  Test: one_letter_a
7  Test passed
8
9  Test: empty_file
10 Test passed
11
12 Test: no_palindroms
13 Test passed
14
15 Test: todos_palindromos
16 Test passed
```

```
17
18 Test: varias_lineas
19 Test passed
20
21 Test: all_letters
22 Test passed
23
24 Test: case_sensitive
25 Test passed
26
27 Test: numbers_and_letters
28 Test passed
29
30 Test: text_with_dash
31 Test passed
32
33 --------------------
34 All 9 tests passed!!!
35 --------------------
```

## 6. Código en C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include <ctype.h>
5  #include <unistd.h>
6  #include <getopt.h>
7  #include <errno.h>
8  #include <string.h>
9
10 int N = 100;
11
12 typedef struct {
13   char *array;
14   size_t used;
15   size_t size;
16   size_t initial_size;
17 } WordArray;
18
19 void init_array(WordArray *a, size_t initial_size){
20     a->array = (char*)malloc(sizeof(char*)*initial_size);
21     if (a->array == NULL){
22         fprintf(stderr, "Error while allocating memory at
               init_array\n");
23         exit(1);
24     }
25     a->used = 0;
26     a->size = initial_size;
27     a->initial_size = initial_size;
28     a->array[0] = '\0';
```

```
29 }
30
31 void clear_array(WordArray *a){
32     free(a->array);
33     a->array = (char*)malloc(sizeof(char*)*a->initial_size);
34     if (a->array == NULL){
35         fprintf(stderr, "Error while allocating memory at
               clear_array\n");
36         exit(1);
37     }
38     a->used = 0;
39     a->size = a->initial_size;
40     a->array[0] = '\0';
41 }
42
43 void insert_char(WordArray *a, char c){
44     if (a->used == a->size){
45         size_t new_size = a->size*2;
46         char *new_array;
47         new_array = (char*)realloc(a->array, sizeof(char*)*
               new_size);
48         if (new_array != NULL){
49             a->array = new_array;
50         }
51         else{
52             free(a->array);
53             fprintf(stderr, "Error reallocating memory\n");
54             exit(1);
55         }
56         a->size = new_size;
57     }
58     a->array[a->used]=c;
59     a->used++;
60 }
61
62 void free_array(WordArray *a){
63     free(a->array);
64     a->array = NULL;
65     a->size = a->used = 0;
66 }
67
68 /* imprimir el uso de tp0 */
69 void print_usage() {
70     printf("Usage: tp0 -i [input_file] -o [output_file]\n");
71 }
72
73 /* imprimir la pagina de ayuda */
74
75 void print_help() {
76     printf("\tUsage:\n"
77       "\t\ttp0 -h\n"
78       "\t\ttp0 -V\n"
79       "\t\ttp0 [options]\n"
80       "\tOptions:\n"
```

```
81        "\t\t-V, --version\tPrint version and quit.\n"
82        "\t\t-h, --help\tPrint this information.\n"
83        "\t\t-i, --input\tLocation of the input file.\n"
84        "\t\t-o, --output\tLocation of the output file.\n"
85        "\tExamples:\n"
86        "\t\ttp0 -i ~/input -o ~/output\n");
87  }
88
89  /* imprimir la version del programa */
90  void print_version(){
91      printf("tp0 2.0\n");
92  }
93
94  /* funcion para determinar si una palabra es capicua o no */
95  int es_capicua(WordArray *word){
96      size_t len = word->used-1;
97      if (len == 0) return 0;
98
99      int capicua = 1;
100     int i=0;
101     while (capicua && i < (len / 2)){
102         if (tolower(word->array[i]) != tolower(word->array[len
              - i - 1])){
103             return 0;
104         }
105         i++;
106     }
107     return 1;
108 }
109
110 /* Lee la palabra de un archivo y la devuelve en word */
111 int read_word (FILE *f, WordArray *word) {
112     int c = fgetc(f);
113     if (c == EOF){
114         if (ferror(f) != 0){
115             fprintf(stderr, "Error leyendo caracter\n");
116             exit(1);
117         }
118         return 0;
119     }
120     while (1){
121         if ( (65 <= c && c <= 90) || //letras mayusculas
122              (97 <= c && c <= 122) || //letras minusculas
123              (48 <= c && c <= 57) || //numeros
124              c == 95 || c == 45){ //barras
125             insert_char(word,c);
126         }else{
127             if (ferror(f) != 0){
128                 fprintf(stderr, "Error leyendo caracter\n");
129                 exit(1);
130             }
131             insert_char(word,'\0');
132             return 1;
133         }
```

```
134         c = fgetc(f);
135     }
136     return 0;
137 }
138 
139 int main(int argc, char *argv[]) {
140 
141     int opt= 0;
142 
143     int help = -1;
144     int version = -1;
145     int input = -1;
146     int output =-1;
147 
148     char *input_filename = NULL;
149     char *output_filename = NULL;
150 
151     // especificacion de las opciones
152     static struct option long_options[] = {
153         {"help",     no_argument,       0,  'h' },
154         {"version",  no_argument,       0,  'V' },
155         {"input",    required_argument, 0,  'i' },
156         {"output",   required_argument, 0,  'o' },
157         {0,          0,                 0,  0   }
158     };
159 
160     int long_index = 0;
161 
162     // evaluacion de los parametros enviados al programa
163     while ((opt = getopt_long(argc, argv,"hVui:o:",
164                    long_options, &long_index )) != -1) {
165         switch (opt) {
166             case 'h' :
167                 help = 0;
168                 break;
169             case 'V' :
170                 version = 0;
171                 break;
172             case 'i' :
173                 input = 0;
174                 input_filename = optarg;
175                 break;
176             case 'o' :
177                 output = 0;
178                 output_filename = optarg;
179                 break;
180             case '?':
181               exit(1);
182             default:
183                 print_usage();
184                 exit(EXIT_FAILURE);
185         }
186     }
187 
```

```
188     // procesamiento de los parametros
189     if (help == 0) {
190         print_help();
191         exit(0);
192     }
193     else if (version == 0) {
194         print_version();
195         exit(0);
196     }
197
198     /* Si no se recibe parametro de ayuda o version se ejecuta
            el programa */
199
200     // estableciendo los archivos de entrada y salida
201     FILE *input_file = stdin;
202     FILE *output_file = stdout;
203
204     if (input == 0){
205         input_file = fopen(input_filename,"r");
206         if (input_file == NULL) {
207             printf ("can't open input file, errno = %d\n",
                        errno);
208             return 1;
209         }
210     }
211     if (output == 0){
212         output_file = fopen(output_filename,"w");
213         if (output_file == NULL) {
214             printf ("Can't open output file, errno = %d\n",
                        errno);
215             return 1;
216         }
217     }
218
219     /* ejecucion del programa */
220
221     WordArray word;
222     init_array(&word,N);
223     int i = read_word(input_file, &word);
224     while (i == 1){
225         if (es_capicua(&word)){
226           int e = fprintf(output_file,"%s\n", word.array);
227             if (e < 0){
228                 fprintf(stderr, "Error writing at file\n");
229                 exit(1);
230             }
231         }
232         clear_array(&word);
233         i = read_word(input_file, &word);
234     }
235     free_array(&word);
236
237     // cierro los archivos
238
```

```
239     if (input == 0){
240         fclose(input_file);
241     }
242     if (output == 0){
243         fclose(output_file);
244     }
245 
246     return 0;
247 }
```

# 7. Código en MIPS

```
1   .file 1 "tp0_3.c"
2   .section .mdebug.abi32
3   .previous
4   .abicalls
5   .globl  N
6   .data
7   .align  2
8   .type N, @object
9   .size N, 4
10 N:
11  .word 100
12  .rdata
13  .align  2
14 $LC0:
15  .ascii  "Error while allocating memory at init_array\n\000"
16  .text
17  .align  2
18  .globl  init_array
19  .ent  init_array
20 init_array:
21  .frame  $fp,40,$ra     # vars= 0, regs= 4/0, args= 16, extra=
        8
22  .mask 0xd0010000,-4
23  .fmask  0x00000000,0
24  .set  noreorder
25  .cpload $t9
26  .set  reorder
27  subu  $sp,$sp,40
28  .cprestore 16
29  sw  $ra,36($sp)
30  sw  $fp,32($sp)
31  sw  $gp,28($sp)
32  sw  $s0,24($sp)
33  move  $fp,$sp
34  sw  $a0,40($fp)
35  sw  $a1,44($fp)
36  lw  $s0,40($fp)
37  lw  $v0,44($fp)
38  sll $v0,$v0,2
```

```
39    move  $a0,$v0
40    la  $t9,malloc
41    jal $ra,$t9
42    sw  $v0,0($s0)
43    lw  $v0,40($fp)
44    lw  $v0,0($v0)
45    bne $v0,$zero,$L18
46    la  $a0,__sF+176
47    la  $a1,$LC0
48    la  $t9,fprintf
49    jal $ra,$t9
50    li  $a0,1     # 0x1
51    la  $t9,exit
52    jal $ra,$t9
53  $L18:
54    lw  $v0,40($fp)
55    sw  $zero,4($v0)
56    lw  $v1,40($fp)
57    lw  $v0,44($fp)
58    sw  $v0,8($v1)
59    lw  $v1,40($fp)
60    lw  $v0,44($fp)
61    sw  $v0,12($v1)
62    lw  $v0,40($fp)
63    lw  $v0,0($v0)
64    sb  $zero,0($v0)
65    move  $sp,$fp
66    lw  $ra,36($sp)
67    lw  $fp,32($sp)
68    lw  $s0,24($sp)
69    addu  $sp,$sp,40
70    j $ra
71    .end  init_array
72    .size init_array, .-init_array
73    .rdata
74    .align  2
75  $LC1:
76    .ascii  "Error while allocating memory at clear_array\n\000"
77    .text
78    .align  2
79    .globl  clear_array
80    .ent  clear_array
81  clear_array:
82    .frame  $fp,40,$ra    # vars= 0, regs= 4/0, args= 16, extra=
        8
83    .mask 0xd0010000,-4
84    .fmask  0x00000000,0
85    .set  noreorder
86    .cpload $t9
87    .set  reorder
88    subu  $sp,$sp,40
89    .cprestore 16
90    sw  $ra,36($sp)
91    sw  $fp,32($sp)
```

```
92    sw   $gp,28($sp)
93    sw   $s0,24($sp)
94    move   $fp,$sp
95    sw   $a0,40($fp)
96    lw   $v0,40($fp)
97    lw   $a0,0($v0)
98    la   $t9,free
99    jal $ra,$t9
100   lw   $s0,40($fp)
101   lw   $v0,40($fp)
102   lw   $v0,12($v0)
103   sll $v0,$v0,2
104   move   $a0,$v0
105   la   $t9,malloc
106   jal $ra,$t9
107   sw   $v0,0($s0)
108   lw   $v0,40($fp)
109   lw   $v0,0($v0)
110   bne $v0,$zero,$L20
111   la   $a0,__sF+176
112   la   $a1,$LC1
113   la   $t9,fprintf
114   jal $ra,$t9
115   li   $a0,1      # 0x1
116   la   $t9,exit
117   jal $ra,$t9
118 $L20:
119   lw   $v0,40($fp)
120   sw   $zero,4($v0)
121   lw   $v1,40($fp)
122   lw   $v0,40($fp)
123   lw   $v0,12($v0)
124   sw   $v0,8($v1)
125   lw   $v0,40($fp)
126   lw   $v0,0($v0)
127   sb   $zero,0($v0)
128   move   $sp,$fp
129   lw   $ra,36($sp)
130   lw   $fp,32($sp)
131   lw   $s0,24($sp)
132   addu   $sp,$sp,40
133   j $ra
134   .end   clear_array
135   .size clear_array, .-clear_array
136   .rdata
137   .align   2
138 $LC2:
139   .ascii   "Error reallocating memory\n\000"
140   .text
141   .align   2
142   .globl   insert_char
143   .ent   insert_char
144 insert_char:
```

```
145   .frame  $fp,56,$ra    # vars= 16, regs= 3/0, args= 16, extra=
          8
146   .mask 0xd0000000,-8
147   .fmask  0x00000000,0
148   .set  noreorder
149   .cpload $t9
150   .set  reorder
151   subu  $sp,$sp,56
152   .cprestore 16
153   sw  $ra,48($sp)
154   sw  $fp,44($sp)
155   sw  $gp,40($sp)
156   move  $fp,$sp
157   sw  $a0,56($fp)
158   move  $v0,$a1
159   sb  $v0,24($fp)
160   lw  $v0,56($fp)
161   lw  $v1,56($fp)
162   lw  $a0,4($v0)
163   lw  $v0,8($v1)
164   bne $a0,$v0,$L22
165   lw  $v0,56($fp)
166   lw  $v0,8($v0)
167   sll $v0,$v0,1
168   sw  $v0,28($fp)
169   lw  $v1,56($fp)
170   lw  $v0,28($fp)
171   sll $v0,$v0,2
172   lw  $a0,0($v1)
173   move  $a1,$v0
174   la  $t9,realloc
175   jal $ra,$t9
176   sw  $v0,32($fp)
177   lw  $v0,32($fp)
178   beq $v0,$zero,$L23
179   lw  $v1,56($fp)
180   lw  $v0,32($fp)
181   sw  $v0,0($v1)
182   b $L24
183 $L23:
184   lw  $v0,56($fp)
185   lw  $a0,0($v0)
186   la  $t9,free
187   jal $ra,$t9
188   la  $a0,__sF+176
189   la  $a1,$LC2
190   la  $t9,fprintf
191   jal $ra,$t9
192   li  $a0,1     # 0x1
193   la  $t9,exit
194   jal $ra,$t9
195 $L24:
196   lw  $v1,56($fp)
197   lw  $v0,28($fp)
```

```
198   sw  $v0,8($v1)
199 $L22:
200   lw  $v0,56($fp)
201   lw  $v1,56($fp)
202   lw  $a0,0($v0)
203   lw  $v0,4($v1)
204   addu  $v1,$a0,$v0
205   lbu $v0,24($fp)
206   sb  $v0,0($v1)
207   lw  $v1,56($fp)
208   lw  $v0,56($fp)
209   lw  $v0,4($v0)
210   addu  $v0,$v0,1
211   sw  $v0,4($v1)
212   move  $sp,$fp
213   lw  $ra,48($sp)
214   lw  $fp,44($sp)
215   addu  $sp,$sp,56
216   j $ra
217   .end  insert_char
218   .size insert_char, .-insert_char
219   .align  2
220   .globl  free_array
221   .ent  free_array
222 free_array:
223   .frame  $fp,40,$ra    # vars= 0, regs= 3/0, args= 16, extra=
        8
224   .mask 0xd0000000,-8
225   .fmask  0x00000000,0
226   .set  noreorder
227   .cpload $t9
228   .set  reorder
229   subu  $sp,$sp,40
230   .cprestore 16
231   sw  $ra,32($sp)
232   sw  $fp,28($sp)
233   sw  $gp,24($sp)
234   move  $fp,$sp
235   sw  $a0,40($fp)
236   lw  $v0,40($fp)
237   lw  $a0,0($v0)
238   la  $t9,free
239   jal $ra,$t9
240   lw  $v0,40($fp)
241   sw  $zero,0($v0)
242   lw  $v1,40($fp)
243   lw  $v0,40($fp)
244   sw  $zero,4($v0)
245   sw  $zero,8($v1)
246   move  $sp,$fp
247   lw  $ra,32($sp)
248   lw  $fp,28($sp)
249   addu  $sp,$sp,40
250   j $ra
```

```
251   .end  free_array
252   .size free_array, .-free_array
253   .rdata
254   .align  2
255 $LC3:
256   .ascii  "Usage: tp0 -i [input_file] -o [output_file]\n\000"
257   .text
258   .align  2
259   .globl  print_usage
260   .ent  print_usage
261 print_usage:
262   .frame  $fp,40,$ra    # vars= 0, regs= 3/0, args= 16, extra=
        8
263   .mask 0xd0000000,-8
264   .fmask  0x00000000,0
265   .set  noreorder
266   .cpload $t9
267   .set  reorder
268   subu  $sp,$sp,40
269   .cprestore 16
270   sw  $ra,32($sp)
271   sw  $fp,28($sp)
272   sw  $gp,24($sp)
273   move  $fp,$sp
274   la  $a0,$LC3
275   la  $t9,printf
276   jal $ra,$t9
277   move  $sp,$fp
278   lw  $ra,32($sp)
279   lw  $fp,28($sp)
280   addu  $sp,$sp,40
281   j $ra
282   .end  print_usage
283   .size print_usage, .-print_usage
284   .rdata
285   .align  2
286 $LC4:
287   .ascii  "\tUsage:\n"
288   .ascii  "\t\ttp0 -h\n"
289   .ascii  "\t\ttp0 -V\n"
290   .ascii  "\t\ttp0 [options]\n"
291   .ascii  "\tOptions:\n"
292   .ascii  "\t\t-V, --version\tPrint version and quit.\n"
293   .ascii  "\t\t-h, --help\tPrint this information.\n"
294   .ascii  "\t\t-i, --input\tLocation of the input file.\n"
295   .ascii  "\t\t-o, --output\tLocation of the output file.\n"
296   .ascii  "\tExamples:\n"
297   .ascii  "\t\ttp0 -i ~/input -o ~/output\n\000"
298   .text
299   .align  2
300   .globl  print_help
301   .ent  print_help
302 print_help:
```

```
303   .frame  $fp,40,$ra     # vars= 0, regs= 3/0, args= 16, extra=
          8
304   .mask 0xd0000000,-8
305   .fmask  0x00000000,0
306   .set  noreorder
307   .cpload $t9
308   .set  reorder
309   subu  $sp,$sp,40
310   .cprestore 16
311   sw  $ra,32($sp)
312   sw  $fp,28($sp)
313   sw  $gp,24($sp)
314   move  $fp,$sp
315   la  $a0,$LC4
316   la  $t9,printf
317   jal $ra,$t9
318   move  $sp,$fp
319   lw  $ra,32($sp)
320   lw  $fp,28($sp)
321   addu  $sp,$sp,40
322   j $ra
323   .end  print_help
324   .size print_help, .-print_help
325   .rdata
326   .align  2
327 $LC5:
328   .ascii  "tp0 2.0\n\000"
329   .text
330   .align  2
331   .globl  print_version
332   .ent  print_version
333 print_version:
334   .frame  $fp,40,$ra     # vars= 0, regs= 3/0, args= 16, extra=
          8
335   .mask 0xd0000000,-8
336   .fmask  0x00000000,0
337   .set  noreorder
338   .cpload $t9
339   .set  reorder
340   subu  $sp,$sp,40
341   .cprestore 16
342   sw  $ra,32($sp)
343   sw  $fp,28($sp)
344   sw  $gp,24($sp)
345   move  $fp,$sp
346   la  $a0,$LC5
347   la  $t9,printf
348   jal $ra,$t9
349   move  $sp,$fp
350   lw  $ra,32($sp)
351   lw  $fp,28($sp)
352   addu  $sp,$sp,40
353   j $ra
354   .end  print_version
```

```
355   .size print_version, .-print_version
356   .align  2
357   .globl  es_capicua
358   .ent  es_capicua
359 es_capicua:
360   .frame  $fp,32,$ra    # vars= 16, regs= 2/0, args= 0, extra=
          8
361   .mask 0x50000000,-4
362   .fmask  0x00000000,0
363   .set  noreorder
364   .cpload $t9
365   .set  reorder
366   subu  $sp,$sp,32
367   .cprestore 0
368   sw  $fp,28($sp)
369   sw  $gp,24($sp)
370   move  $fp,$sp
371   sw  $a0,32($fp)
372   lw  $v0,32($fp)
373   lw  $v0,4($v0)
374   addu  $v0,$v0,-1
375   sw  $v0,8($fp)
376   lw  $v0,8($fp)
377   bne $v0,$zero,$L30
378   sw  $zero,20($fp)
379   b $L29
380 $L30:
381   li  $v0,1     # 0x1
382   sw  $v0,12($fp)
383   sw  $zero,16($fp)
384 $L31:
385   lw  $v0,12($fp)
386   beq $v0,$zero,$L32
387   lw  $v0,8($fp)
388   srl $v1,$v0,1
389   lw  $v0,16($fp)
390   sltu  $v0,$v0,$v1
391   bne $v0,$zero,$L33
392   b $L32
393 $L33:
394   lw  $v0,32($fp)
395   lw  $v1,0($v0)
396   lw  $v0,16($fp)
397   addu  $v0,$v1,$v0
398   lb  $v0,0($v0)
399   sll $v1,$v0,1
400   lw  $v0,_tolower_tab_
401   addu  $v0,$v1,$v0
402   addu  $a1,$v0,2
403   lw  $a0,32($fp)
404   lw  $v1,8($fp)
405   lw  $v0,16($fp)
406   subu  $v1,$v1,$v0
407   lw  $v0,0($a0)
```

```
408   addu  $v0,$v1,$v0
409   addu  $v0,$v0,-1
410   lb  $v0,0($v0)
411   sll $v1,$v0,1
412   lw  $v0,_tolower_tab_
413   addu  $v0,$v1,$v0
414   addu  $v0,$v0,2
415   lh  $v1,0($a1)
416   lh  $v0,0($v0)
417   beq $v1,$v0,$L35
418   sw  $zero,20($fp)
419   b $L29
420 $L35:
421   lw  $v0,16($fp)
422   addu  $v0,$v0,1
423   sw  $v0,16($fp)
424   b $L31
425 $L32:
426   li  $v0,1     # 0x1
427   sw  $v0,20($fp)
428 $L29:
429   lw  $v0,20($fp)
430   move  $sp,$fp
431   lw  $fp,28($sp)
432   addu  $sp,$sp,32
433   j $ra
434   .end  es_capicua
435   .size es_capicua, .-es_capicua
436   .rdata
437   .align  2
438 $LC6:
439   .ascii  "Error leyendo caracter\n\000"
440   .text
441   .align  2
442   .globl  read_word
443   .ent  read_word
444 read_word:
445   .frame  $fp,48,$ra    # vars= 8, regs= 3/0, args= 16, extra=
        8
446   .mask 0xd0000000,-8
447   .fmask  0x00000000,0
448   .set  noreorder
449   .cpload $t9
450   .set  reorder
451   subu  $sp,$sp,48
452   .cprestore 16
453   sw  $ra,40($sp)
454   sw  $fp,36($sp)
455   sw  $gp,32($sp)
456   move  $fp,$sp
457   sw  $a0,48($fp)
458   sw  $a1,52($fp)
459   lw  $a0,48($fp)
460   la  $t9,fgetc
```

```
461   jal $ra,$t9
462   sw  $v0,24($fp)
463   lw  $v1,24($fp)
464   li  $v0,-1      # 0xffffffffffffffff
465   bne $v1,$v0,$L37
466   lw  $v0,48($fp)
467   lhu $v0,12($v0)
468   srl $v0,$v0,6
469   andi  $v0,$v0,0x1
470   beq $v0,$zero,$L38
471   la  $a0,__sF+176
472   la  $a1,$LC6
473   la  $t9,fprintf
474   jal $ra,$t9
475   li  $a0,1     # 0x1
476   la  $t9,exit
477   jal $ra,$t9
478 $L38:
479   sw  $zero,28($fp)
480   b $L36
481 $L37:
482   .set  noreorder
483   nop
484   .set  reorder
485 $L39:
486   lw  $v0,24($fp)
487   slt $v0,$v0,65
488   bne $v0,$zero,$L44
489   lw  $v0,24($fp)
490   slt $v0,$v0,91
491   bne $v0,$zero,$L43
492 $L44:
493   lw  $v0,24($fp)
494   slt $v0,$v0,97
495   bne $v0,$zero,$L45
496   lw  $v0,24($fp)
497   slt $v0,$v0,123
498   bne $v0,$zero,$L43
499 $L45:
500   lw  $v0,24($fp)
501   slt $v0,$v0,48
502   bne $v0,$zero,$L46
503   lw  $v0,24($fp)
504   slt $v0,$v0,58
505   bne $v0,$zero,$L43
506 $L46:
507   lw  $v1,24($fp)
508   li  $v0,95      # 0x5f
509   beq $v1,$v0,$L43
510   lw  $v1,24($fp)
511   li  $v0,45      # 0x2d
512   beq $v1,$v0,$L43
513   b $L42
514 $L43:
```

```
515   lb  $v0,24($fp)
516   lw  $a0,52($fp)
517   move  $a1,$v0
518   la  $t9,insert_char
519   jal $ra,$t9
520   b $L47
521 $L42:
522   lw  $v0,48($fp)
523   lhu $v0,12($v0)
524   srl $v0,$v0,6
525   andi  $v0,$v0,0x1
526   beq $v0,$zero,$L48
527   la  $a0,__sF+176
528   la  $a1,$LC6
529   la  $t9,fprintf
530   jal $ra,$t9
531   li  $a0,1     # 0x1
532   la  $t9,exit
533   jal $ra,$t9
534 $L48:
535   lw  $a0,52($fp)
536   move  $a1,$zero
537   la  $t9,insert_char
538   jal $ra,$t9
539   li  $v0,1     # 0x1
540   sw  $v0,28($fp)
541   b $L36
542 $L47:
543   lw  $a0,48($fp)
544   la  $t9,fgetc
545   jal $ra,$t9
546   sw  $v0,24($fp)
547   b $L39
548 $L36:
549   lw  $v0,28($fp)
550   move  $sp,$fp
551   lw  $ra,40($sp)
552   lw  $fp,36($sp)
553   addu  $sp,$sp,48
554   j $ra
555   .end  read_word
556   .size read_word, .-read_word
557   .rdata
558   .align  2
559 $LC7:
560   .ascii  "help\000"
561   .align  2
562 $LC8:
563   .ascii  "version\000"
564   .align  2
565 $LC9:
566   .ascii  "input\000"
567   .align  2
568 $LC10:
```

```
569   .ascii  "output\000"
570   .data
571   .align  2
572   .type long_options.0, @object
573   .size long_options.0, 80
574 long_options.0:
575   .word $LC7
576   .word 0
577   .word 0
578   .word 104
579   .word $LC8
580   .word 0
581   .word 0
582   .word 86
583   .word $LC9
584   .word 1
585   .word 0
586   .word 105
587   .word $LC10
588   .word 1
589   .word 0
590   .word 111
591   .word 0
592   .word 0
593   .word 0
594   .word 0
595   .rdata
596   .align  2
597 $LC11:
598   .ascii  "hVui:o:\000"
599   .align  2
600 $LC12:
601   .ascii  "r\000"
602   .align  2
603 $LC13:
604   .ascii  "can't open input file, errno = %d\n\000"
605   .align  2
606 $LC14:
607   .ascii  "w\000"
608   .align  2
609 $LC15:
610   .ascii  "Can't open output file, errno = %d\n\000"
611   .align  2
612 $LC16:
613   .ascii  "%s\n\000"
614   .align  2
615 $LC17:
616   .ascii  "Error writing at file\n\000"
617   .text
618   .align  2
619   .globl  main
620   .ent  main
621 main:
```

```
622   .frame  $fp,120,$ra   # vars= 72, regs= 3/0, args= 24, extra=
          8
623   .mask 0xd0000000,-8
624   .fmask  0x00000000,0
625   .set  noreorder
626   .cpload $t9
627   .set  reorder
628   subu  $sp,$sp,120
629   .cprestore 24
630   sw  $ra,112($sp)
631   sw  $fp,108($sp)
632   sw  $gp,104($sp)
633   move  $fp,$sp
634   sw  $a0,120($fp)
635   sw  $a1,124($fp)
636   sw  $zero,32($fp)
637   li  $v0,-1      # 0xffffffffffffffff
638   sw  $v0,36($fp)
639   li  $v0,-1      # 0xffffffffffffffff
640   sw  $v0,40($fp)
641   li  $v0,-1      # 0xffffffffffffffff
642   sw  $v0,44($fp)
643   li  $v0,-1      # 0xffffffffffffffff
644   sw  $v0,48($fp)
645   sw  $zero,52($fp)
646   sw  $zero,56($fp)
647   sw  $zero,60($fp)
648 $L50:
649   addu  $v0,$fp,60
650   sw  $v0,16($sp)
651   lw  $a0,120($fp)
652   lw  $a1,124($fp)
653   la  $a2,$LC11
654   la  $a3,long_options.0
655   la  $t9,getopt_long
656   jal $ra,$t9
657   sw  $v0,32($fp)
658   lw  $v1,32($fp)
659   li  $v0,-1      # 0xffffffffffffffff
660   bne $v1,$v0,$L52
661   b $L51
662 $L52:
663   lw  $v0,32($fp)
664   addu  $v0,$v0,-63
665   sw  $v0,100($fp)
666   lw  $v1,100($fp)
667   sltu  $v0,$v1,49
668   beq $v0,$zero,$L59
669   lw  $v0,100($fp)
670   sll $v1,$v0,2
671   la  $v0,$L60
672   addu  $v0,$v1,$v0
673   lw  $v0,0($v0)
674   .cpadd  $v0
```

```
675   j $v0
676   .rdata
677   .align  2
678 $L60:
679   .gpword $L58
680   .gpword $L59
681   .gpword $L59
682   .gpword $L59
683   .gpword $L59
684   .gpword $L59
685   .gpword $L59
686   .gpword $L59
687   .gpword $L59
688   .gpword $L59
689   .gpword $L59
690   .gpword $L59
691   .gpword $L59
692   .gpword $L59
693   .gpword $L59
694   .gpword $L59
695   .gpword $L59
696   .gpword $L59
697   .gpword $L59
698   .gpword $L59
699   .gpword $L59
700   .gpword $L59
701   .gpword $L59
702   .gpword $L55
703   .gpword $L59
704   .gpword $L59
705   .gpword $L59
706   .gpword $L59
707   .gpword $L59
708   .gpword $L59
709   .gpword $L59
710   .gpword $L59
711   .gpword $L59
712   .gpword $L59
713   .gpword $L59
714   .gpword $L59
715   .gpword $L59
716   .gpword $L59
717   .gpword $L59
718   .gpword $L59
719   .gpword $L59
720   .gpword $L54
721   .gpword $L56
722   .gpword $L59
723   .gpword $L59
724   .gpword $L59
725   .gpword $L59
726   .gpword $L59
727   .gpword $L57
728   .text
```

```
729 $L54:
730   sw  $zero,36($fp)
731   b $L50
732 $L55:
733   sw  $zero,40($fp)
734   b $L50
735 $L56:
736   sw  $zero,44($fp)
737   lw  $v0,optarg
738   sw  $v0,52($fp)
739   b $L50
740 $L57:
741   sw  $zero,48($fp)
742   lw  $v0,optarg
743   sw  $v0,56($fp)
744   b $L50
745 $L58:
746   li  $a0,1     # 0x1
747   la  $t9,exit
748   jal $ra,$t9
749 $L59:
750   la  $t9,print_usage
751   jal $ra,$t9
752   li  $a0,1     # 0x1
753   la  $t9,exit
754   jal $ra,$t9
755 $L51:
756   lw  $v0,36($fp)
757   bne $v0,$zero,$L61
758   la  $t9,print_help
759   jal $ra,$t9
760   move  $a0,$zero
761   la  $t9,exit
762   jal $ra,$t9
763 $L61:
764   lw  $v0,40($fp)
765   bne $v0,$zero,$L62
766   la  $t9,print_version
767   jal $ra,$t9
768   move  $a0,$zero
769   la  $t9,exit
770   jal $ra,$t9
771 $L62:
772   la  $v0,__sF
773   sw  $v0,64($fp)
774   la  $v0,__sF+88
775   sw  $v0,68($fp)
776   lw  $v0,44($fp)
777   bne $v0,$zero,$L64
778   lw  $a0,52($fp)
779   la  $a1,$LC12
780   la  $t9,fopen
781   jal $ra,$t9
782   sw  $v0,64($fp)
```

```
783   lw  $v0,64($fp)
784   bne $v0,$zero,$L64
785   la  $t9,__errno
786   jal $ra,$t9
787   la  $a0,$LC13
788   lw  $a1,0($v0)
789   la  $t9,printf
790   jal $ra,$t9
791   li  $v0,1     # 0x1
792   sw  $v0,96($fp)
793   b $L49
794 $L64:
795   lw  $v0,48($fp)
796   bne $v0,$zero,$L66
797   lw  $a0,56($fp)
798   la  $a1,$LC14
799   la  $t9,fopen
800   jal $ra,$t9
801   sw  $v0,68($fp)
802   lw  $v0,68($fp)
803   bne $v0,$zero,$L66
804   la  $t9,__errno
805   jal $ra,$t9
806   la  $a0,$LC15
807   lw  $a1,0($v0)
808   la  $t9,printf
809   jal $ra,$t9
810   li  $v1,1     # 0x1
811   sw  $v1,96($fp)
812   b $L49
813 $L66:
814   addu  $v0,$fp,72
815   move  $a0,$v0
816   lw  $a1,N
817   la  $t9,init_array
818   jal $ra,$t9
819   addu  $v0,$fp,72
820   lw  $a0,64($fp)
821   move  $a1,$v0
822   la  $t9,read_word
823   jal $ra,$t9
824   sw  $v0,88($fp)
825 $L68:
826   lw  $v1,88($fp)
827   li  $v0,1     # 0x1
828   beq $v1,$v0,$L70
829   b $L69
830 $L70:
831   addu  $v0,$fp,72
832   move  $a0,$v0
833   la  $t9,es_capicua
834   jal $ra,$t9
835   beq $v0,$zero,$L71
836   lw  $a0,68($fp)
```

```
837   la  $a1,$LC16
838   lw  $a2,72($fp)
839   la  $t9,fprintf
840   jal $ra,$t9
841   sw  $v0,92($fp)
842   lw  $v0,92($fp)
843   bgez  $v0,$L71
844   la  $a0,__sF+176
845   la  $a1,$LC17
846   la  $t9,fprintf
847   jal $ra,$t9
848   li  $a0,1     # 0x1
849   la  $t9,exit
850   jal $ra,$t9
851 $L71:
852   addu  $v0,$fp,72
853   move  $a0,$v0
854   la  $t9,clear_array
855   jal $ra,$t9
856   addu  $v0,$fp,72
857   lw  $a0,64($fp)
858   move  $a1,$v0
859   la  $t9,read_word
860   jal $ra,$t9
861   sw  $v0,88($fp)
862   b $L68
863 $L69:
864   addu  $v0,$fp,72
865   move  $a0,$v0
866   la  $t9,free_array
867   jal $ra,$t9
868   lw  $v0,44($fp)
869   bne $v0,$zero,$L73
870   lw  $a0,64($fp)
871   la  $t9,fclose
872   jal $ra,$t9
873 $L73:
874   lw  $v0,48($fp)
875   bne $v0,$zero,$L74
876   lw  $a0,68($fp)
877   la  $t9,fclose
878   jal $ra,$t9
879 $L74:
880   sw  $zero,96($fp)
881 $L49:
882   lw  $v0,96($fp)
883   move  $sp,$fp
884   lw  $ra,112($sp)
885   lw  $fp,108($sp)
886   addu  $sp,$sp,120
887   j $ra
888   .end  main
889   .size main, .-main
890   .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

# 8. Enunciado

El enunciado se encuentra anexado al final de este documento.