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     a->used = 0;
22     a->size = initial_size;
23     a->initial_size = initial_size;
24     a->array[0] = '\0';
25 }
26
27 void clear_array(WordArray *a){
28     free(a->array);
29     a->array = (char*)malloc(sizeof(char*)*a->initial_size);
```

```
30      a->used = 0;
31      a->size = a->initial_size;
32      a->array[0] = '\0';
33  }
34
35  void insert_char(WordArray *a, char c){
36      if (a->used == a->size){
37          size_t new_size = a->size*2;
38          a->array = (char*)realloc(a->array, sizeof(char*)*
                new_size);
39          a->size = new_size;
40      }
41      a->array[a->used]=c;
42      a->array[a->used+1]='\0';
43      a->used++;
44  }
45
46  void free_array(WordArray *a){
47      free(a->array);
48      a->array = NULL;
49      a->size = a->used = 0;
50  }
51
52  /* imprimir el uso de tp0 */
53  void print_usage() {
54      printf("Usage: tp0 -i [input_file] -o [output_file]\n");
55  }
56
57  /* imprimir la pagina de ayuda */
58
59  void print_help() {
60      printf("\tUsage:\n"
61        "\t\ttp0 -h\n"
62        "\t\ttp0 -V\n"
63        "\t\ttp0 [options]\n"
64        "\tOptions:\n"
65        "\t\t-V, --version\tPrint version and quit.\n"
66        "\t\t-h, --help\tPrint this information.\n"
67        "\t\t-i, --input\tLocation of the input file.\n"
68        "\t\t-o, --output\tLocation of the output file.\n"
69        "\tExamples:\n"
70        "\t\ttp0 -i ~/input -o ~/output\n");
71  }
72
73  /* imprimir la version del programa */
74  void print_version(){
75      printf("tp0 2.0\n");
76  }
77
78  /* funcion para determinar si una palabra es capicua o no */
79  int es_capicua(WordArray *word){
80
81      size_t len = word->used;
82      if (len == 0) return 0;
```

```
83
84      int capicua = 1;
85      int i=0;
86      while (capicua && i < (len / 2)){
87          if (tolower(word->array[i]) != tolower(word->array[len
                - i - 1])){
88              return 0;
89          }
90          i++;
91      }
92      return 1;
93  }
94
95  /* Lee la palabra de un archivo y la devuelve en word */
96  int read_word (FILE *f, WordArray *word) {
97      int c = fgetc(f);
98      if (c == EOF) return 0;
99      while (1){
100         if ( (65 <= c && c <= 90) || //letras mayusculas
101              (97 <= c && c <= 122) || //letras minusculas
102              (48 <= c && c <= 57) || //numeros
103              c == 95 || c == 45){ //barras
104             insert_char(word,c);
105         }else{
106             return 1;
107         }
108         c = fgetc(f);
109     }
110     return 0;
111 }
112
113 int main(int argc, char *argv[]) {
114
115     int opt= 0;
116
117     int help = -1;
118     int version = -1;
119     int input = -1;
120     int output =-1;
121
122     char *input_filename = NULL;
123     char *output_filename = NULL;
124
125     // especificacion de las opciones
126     static struct option long_options[] = {
127         {"help",     no_argument,       0,  'h' },
128         {"version",  no_argument,       0,  'V' },
129         {"input",    required_argument, 0,  'i' },
130         {"output",   required_argument, 0,  'o' },
131         {0,          0,                 0,  0   }
132     };
133
134     int long_index = 0;
135
```

```
136     // evaluacion de los parametros enviados al programa
137     while ((opt = getopt_long(argc, argv,"hVui:o:",
138                     long_options, &long_index )) != -1) {
139         switch (opt) {
140             case 'h' :
141                 help = 0;
142                 break;
143             case 'V' :
144                 version = 0;
145                 break;
146             case 'i' :
147                 input = 0;
148                 input_filename = optarg;
149                 break;
150             case 'o' :
151                 output = 0;
152                 output_filename = optarg;
153                 break;
154             case '?':
155               exit(1);
156             default:
157                 print_usage();
158                 exit(EXIT_FAILURE);
159         }
160     }
161
162     // procesamiento de los parametros
163     if (help == 0) {
164         print_help();
165         exit(0);
166     }
167     else if (version == 0) {
168         print_version();
169         exit(0);
170     }
171
172     /* Si no se recibe parametro de ayuda o version se ejecuta
        el programa */
173
174     // estableciendo los archivos de entrada y salida
175     FILE *input_file = stdin;
176     FILE *output_file = stdout;
177
178     if (input == 0){
179         input_file = fopen(input_filename,"r");
180         if (input_file == NULL) {
181             printf ("can't open input file, errno = %d\n",
                    errno);
182             return 1;
183         }
184     }
185     if (output == 0){
186         output_file = fopen(output_filename,"w");
187         if (output_file == NULL) {
```

```
188             printf ("Can't open output file, errno = %d\n",
                    errno);
189             return 1;
190         }
191     }
192
193     /* ejecucion del programa */
194
195     WordArray word;
196     init_array(&word,N);
197     int i = read_word(input_file, &word);
198     while (i == 1){
199         if (es_capicua(&word)){
200           fprintf(output_file,"%s\n", word.array);
201         }
202         clear_array(&word);
203         i = read_word(input_file, &word);
204     }
205     free_array(&word);
206
207     // cierro los archivos
208
209     if (input == 0){
210         fclose(input_file);
211     }
212     if (output == 0){
213         fclose(output_file);
214     }
215
216     return 0;
217 }
```

7. Código en MIPS

```
1   .file 1 "tp0_2.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  .text
13  .align  2
14  .globl  init_array
15  .ent  init_array
16 init_array:
```

```
17    .frame  $fp,40,$ra     # vars= 0, regs= 4/0, args= 16, extra=
          8
18    .mask 0xd0010000,-4
19    .fmask  0x00000000,0
20    .set  noreorder
21    .cpload $t9
22    .set  reorder
23    subu  $sp,$sp,40
24    .cprestore 16
25    sw  $ra,36($sp)
26    sw  $fp,32($sp)
27    sw  $gp,28($sp)
28    sw  $s0,24($sp)
29    move  $fp,$sp
30    sw  $a0,40($fp)
31    sw  $a1,44($fp)
32    lw  $s0,40($fp)
33    lw  $v0,44($fp)
34    sll $v0,$v0,2
35    move  $a0,$v0
36    la  $t9,malloc
37    jal $ra,$t9
38    sw  $v0,0($s0)
39    lw  $v0,40($fp)
40    sw  $zero,4($v0)
41    lw  $v1,40($fp)
42    lw  $v0,44($fp)
43    sw  $v0,8($v1)
44    lw  $v1,40($fp)
45    lw  $v0,44($fp)
46    sw  $v0,12($v1)
47    lw  $v0,40($fp)
48    lw  $v0,0($v0)
49    sb  $zero,0($v0)
50    move  $sp,$fp
51    lw  $ra,36($sp)
52    lw  $fp,32($sp)
53    lw  $s0,24($sp)
54    addu  $sp,$sp,40
55    j $ra
56    .end  init_array
57    .size init_array, .-init_array
58    .align  2
59    .globl  clear_array
60    .ent  clear_array
61  clear_array:
62    .frame  $fp,40,$ra     # vars= 0, regs= 4/0, args= 16, extra=
          8
63    .mask 0xd0010000,-4
64    .fmask  0x00000000,0
65    .set  noreorder
66    .cpload $t9
67    .set  reorder
68    subu  $sp,$sp,40
```

```
69    .cprestore 16
70    sw  $ra,36($sp)
71    sw  $fp,32($sp)
72    sw  $gp,28($sp)
73    sw  $s0,24($sp)
74    move  $fp,$sp
75    sw  $a0,40($fp)
76    lw  $v0,40($fp)
77    lw  $a0,0($v0)
78    la  $t9,free
79    jal $ra,$t9
80    lw  $s0,40($fp)
81    lw  $v0,40($fp)
82    lw  $v0,12($v0)
83    sll $v0,$v0,2
84    move  $a0,$v0
85    la  $t9,malloc
86    jal $ra,$t9
87    sw  $v0,0($s0)
88    lw  $v0,40($fp)
89    sw  $zero,4($v0)
90    lw  $v1,40($fp)
91    lw  $v0,40($fp)
92    lw  $v0,12($v0)
93    sw  $v0,8($v1)
94    lw  $v0,40($fp)
95    lw  $v0,0($v0)
96    sb  $zero,0($v0)
97    move  $sp,$fp
98    lw  $ra,36($sp)
99    lw  $fp,32($sp)
100   lw  $s0,24($sp)
101   addu  $sp,$sp,40
102   j $ra
103   .end  clear_array
104   .size clear_array, .-clear_array
105   .align  2
106   .globl  insert_char
107   .ent  insert_char
108 insert_char:
109   .frame  $fp,48,$ra    # vars= 8, regs= 4/0, args= 16, extra=
          8
110   .mask 0xd0010000,-4
111   .fmask  0x00000000,0
112   .set  noreorder
113   .cpload $t9
114   .set  reorder
115   subu  $sp,$sp,48
116   .cprestore 16
117   sw  $ra,44($sp)
118   sw  $fp,40($sp)
119   sw  $gp,36($sp)
120   sw  $s0,32($sp)
121   move  $fp,$sp
```

```
122   sw  $a0,48($fp)
123   move  $v0,$a1
124   sb  $v0,24($fp)
125   lw  $v0,48($fp)
126   lw  $v1,48($fp)
127   lw  $a0,4($v0)
128   lw  $v0,8($v1)
129   bne $a0,$v0,$L20
130   lw  $v0,48($fp)
131   lw  $v0,8($v0)
132   sll $v0,$v0,1
133   sw  $v0,28($fp)
134   lw  $s0,48($fp)
135   lw  $v1,48($fp)
136   lw  $v0,28($fp)
137   sll $v0,$v0,2
138   lw  $a0,0($v1)
139   move  $a1,$v0
140   la  $t9,realloc
141   jal $ra,$t9
142   sw  $v0,0($s0)
143   lw  $v1,48($fp)
144   lw  $v0,28($fp)
145   sw  $v0,8($v1)
146 $L20:
147   lw  $v0,48($fp)
148   lw  $v1,48($fp)
149   lw  $a0,0($v0)
150   lw  $v0,4($v1)
151   addu  $v1,$a0,$v0
152   lbu $v0,24($fp)
153   sb  $v0,0($v1)
154   lw  $v0,48($fp)
155   lw  $v1,48($fp)
156   lw  $a0,0($v0)
157   lw  $v0,4($v1)
158   addu  $v0,$a0,$v0
159   addu  $v0,$v0,1
160   sb  $zero,0($v0)
161   lw  $v1,48($fp)
162   lw  $v0,48($fp)
163   lw  $v0,4($v0)
164   addu  $v0,$v0,1
165   sw  $v0,4($v1)
166   move  $sp,$fp
167   lw  $ra,44($sp)
168   lw  $fp,40($sp)
169   lw  $s0,32($sp)
170   addu  $sp,$sp,48
171   j $ra
172   .end  insert_char
173   .size insert_char, .-insert_char
174   .align  2
175   .globl  free_array
```

```
176   .ent  free_array
177 free_array:
178   .frame  $fp,40,$ra     # vars= 0, regs= 3/0, args= 16, extra=
          8
179   .mask 0xd0000000,-8
180   .fmask  0x00000000,0
181   .set  noreorder
182   .cpload $t9
183   .set  reorder
184   subu  $sp,$sp,40
185   .cprestore 16
186   sw  $ra,32($sp)
187   sw  $fp,28($sp)
188   sw  $gp,24($sp)
189   move  $fp,$sp
190   sw  $a0,40($fp)
191   lw  $v0,40($fp)
192   lw  $a0,0($v0)
193   la  $t9,free
194   jal $ra,$t9
195   lw  $v0,40($fp)
196   sw  $zero,0($v0)
197   lw  $v1,40($fp)
198   lw  $v0,40($fp)
199   sw  $zero,4($v0)
200   sw  $zero,8($v1)
201   move  $sp,$fp
202   lw  $ra,32($sp)
203   lw  $fp,28($sp)
204   addu  $sp,$sp,40
205   j $ra
206   .end  free_array
207   .size free_array, .-free_array
208   .rdata
209   .align  2
210 $LC0:
211   .ascii  "Usage: tp0 -i [input_file] -o [output_file]\n\000"
212   .text
213   .align  2
214   .globl  print_usage
215   .ent  print_usage
216 print_usage:
217   .frame  $fp,40,$ra     # vars= 0, regs= 3/0, args= 16, extra=
          8
218   .mask 0xd0000000,-8
219   .fmask  0x00000000,0
220   .set  noreorder
221   .cpload $t9
222   .set  reorder
223   subu  $sp,$sp,40
224   .cprestore 16
225   sw  $ra,32($sp)
226   sw  $fp,28($sp)
227   sw  $gp,24($sp)
```

```
228   move  $fp,$sp
229   la  $a0,$LC0
230   la  $t9,printf
231   jal $ra,$t9
232   move  $sp,$fp
233   lw  $ra,32($sp)
234   lw  $fp,28($sp)
235   addu  $sp,$sp,40
236   j $ra
237   .end  print_usage
238   .size print_usage, .-print_usage
239   .rdata
240   .align  2
241 $LC1:
242   .ascii  "\tUsage:\n"
243   .ascii  "\t\ttp0 -h\n"
244   .ascii  "\t\ttp0 -V\n"
245   .ascii  "\t\ttp0 [options]\n"
246   .ascii  "\tOptions:\n"
247   .ascii  "\t\t-V, --version\tPrint version and quit.\n"
248   .ascii  "\t\t-h, --help\tPrint this information.\n"
249   .ascii  "\t\t-i, --input\tLocation of the input file.\n"
250   .ascii  "\t\t-o, --output\tLocation of the output file.\n"
251   .ascii  "\tExamples:\n"
252   .ascii  "\t\ttp0 -i ~/input -o ~/output\n\000"
253   .text
254   .align  2
255   .globl  print_help
256   .ent  print_help
257 print_help:
258   .frame  $fp,40,$ra    # vars= 0, regs= 3/0, args= 16, extra=
        8
259   .mask 0xd0000000,-8
260   .fmask  0x00000000,0
261   .set  noreorder
262   .cpload $t9
263   .set  reorder
264   subu  $sp,$sp,40
265   .cprestore 16
266   sw  $ra,32($sp)
267   sw  $fp,28($sp)
268   sw  $gp,24($sp)
269   move  $fp,$sp
270   la  $a0,$LC1
271   la  $t9,printf
272   jal $ra,$t9
273   move  $sp,$fp
274   lw  $ra,32($sp)
275   lw  $fp,28($sp)
276   addu  $sp,$sp,40
277   j $ra
278   .end  print_help
279   .size print_help, .-print_help
280   .rdata
```

```
281   .align  2
282 $LC2:
283   .ascii  "tp0 2.0\n\000"
284   .text
285   .align  2
286   .globl  print_version
287   .ent  print_version
288 print_version:
289   .frame  $fp,40,$ra     # vars= 0, regs= 3/0, args= 16, extra=
          8
290   .mask 0xd0000000,-8
291   .fmask  0x00000000,0
292   .set  noreorder
293   .cpload $t9
294   .set  reorder
295   subu  $sp,$sp,40
296   .cprestore 16
297   sw  $ra,32($sp)
298   sw  $fp,28($sp)
299   sw  $gp,24($sp)
300   move  $fp,$sp
301   la  $a0,$LC2
302   la  $t9,printf
303   jal $ra,$t9
304   move  $sp,$fp
305   lw  $ra,32($sp)
306   lw  $fp,28($sp)
307   addu  $sp,$sp,40
308   j $ra
309   .end  print_version
310   .size print_version, .-print_version
311   .align  2
312   .globl  es_capicua
313   .ent  es_capicua
314 es_capicua:
315   .frame  $fp,32,$ra     # vars= 16, regs= 2/0, args= 0, extra=
          8
316   .mask 0x50000000,-4
317   .fmask  0x00000000,0
318   .set  noreorder
319   .cpload $t9
320   .set  reorder
321   subu  $sp,$sp,32
322   .cprestore 0
323   sw  $fp,28($sp)
324   sw  $gp,24($sp)
325   move  $fp,$sp
326   sw  $a0,32($fp)
327   lw  $v0,32($fp)
328   lw  $v0,4($v0)
329   sw  $v0,8($fp)
330   lw  $v0,8($fp)
331   bne $v0,$zero,$L26
332   sw  $zero,20($fp)
```

```
333   b $L25
334 $L26:
335   li  $v0,1     # 0x1
336   sw  $v0,12($fp)
337   sw  $zero,16($fp)
338 $L27:
339   lw  $v0,12($fp)
340   beq $v0,$zero,$L28
341   lw  $v0,8($fp)
342   srl $v1,$v0,1
343   lw  $v0,16($fp)
344   sltu  $v0,$v0,$v1
345   bne $v0,$zero,$L29
346   b $L28
347 $L29:
348   lw  $v0,32($fp)
349   lw  $v1,0($v0)
350   lw  $v0,16($fp)
351   addu  $v0,$v1,$v0
352   lb  $v0,0($v0)
353   sll $v1,$v0,1
354   lw  $v0,_tolower_tab_
355   addu  $v0,$v1,$v0
356   addu  $a1,$v0,2
357   lw  $a0,32($fp)
358   lw  $v1,8($fp)
359   lw  $v0,16($fp)
360   subu  $v1,$v1,$v0
361   lw  $v0,0($a0)
362   addu  $v0,$v1,$v0
363   addu  $v0,$v0,-1
364   lb  $v0,0($v0)
365   sll $v1,$v0,1
366   lw  $v0,_tolower_tab_
367   addu  $v0,$v1,$v0
368   addu  $v0,$v0,2
369   lh  $v1,0($a1)
370   lh  $v0,0($v0)
371   beq $v1,$v0,$L31
372   sw  $zero,20($fp)
373   b $L25
374 $L31:
375   lw  $v0,16($fp)
376   addu  $v0,$v0,1
377   sw  $v0,16($fp)
378   b $L27
379 $L28:
380   li  $v0,1     # 0x1
381   sw  $v0,20($fp)
382 $L25:
383   lw  $v0,20($fp)
384   move  $sp,$fp
385   lw  $fp,28($sp)
386   addu  $sp,$sp,32
```

```
387   j $ra
388   .end  es_capicua
389   .size es_capicua, .-es_capicua
390   .align  2
391   .globl  read_word
392   .ent  read_word
393 read_word:
394   .frame  $fp,48,$ra    # vars= 8, regs= 3/0, args= 16, extra=
          8
395   .mask 0xd0000000,-8
396   .fmask  0x00000000,0
397   .set  noreorder
398   .cpload $t9
399   .set  reorder
400   subu  $sp,$sp,48
401   .cprestore 16
402   sw  $ra,40($sp)
403   sw  $fp,36($sp)
404   sw  $gp,32($sp)
405   move  $fp,$sp
406   sw  $a0,48($fp)
407   sw  $a1,52($fp)
408   lw  $a0,48($fp)
409   la  $t9,fgetc
410   jal $ra,$t9
411   sw  $v0,24($fp)
412   lw  $v1,24($fp)
413   li  $v0,-1      # 0xffffffffffffffff
414   bne $v1,$v0,$L33
415   sw  $zero,28($fp)
416   b $L32
417 $L33:
418   .set  noreorder
419   nop
420   .set  reorder
421 $L34:
422   lw  $v0,24($fp)
423   slt $v0,$v0,65
424   bne $v0,$zero,$L39
425   lw  $v0,24($fp)
426   slt $v0,$v0,91
427   bne $v0,$zero,$L38
428 $L39:
429   lw  $v0,24($fp)
430   slt $v0,$v0,97
431   bne $v0,$zero,$L40
432   lw  $v0,24($fp)
433   slt $v0,$v0,123
434   bne $v0,$zero,$L38
435 $L40:
436   lw  $v0,24($fp)
437   slt $v0,$v0,48
438   bne $v0,$zero,$L41
439   lw  $v0,24($fp)
```

```
440   slt $v0,$v0,58
441   bne $v0,$zero,$L38
442 $L41:
443   lw  $v1,24($fp)
444   li  $v0,95      # 0x5f
445   beq $v1,$v0,$L38
446   lw  $v1,24($fp)
447   li  $v0,45      # 0x2d
448   beq $v1,$v0,$L38
449   b $L37
450 $L38:
451   lb  $v0,24($fp)
452   lw  $a0,52($fp)
453   move  $a1,$v0
454   la  $t9,insert_char
455   jal $ra,$t9
456   b $L42
457 $L37:
458   li  $v0,1     # 0x1
459   sw  $v0,28($fp)
460   b $L32
461 $L42:
462   lw  $a0,48($fp)
463   la  $t9,fgetc
464   jal $ra,$t9
465   sw  $v0,24($fp)
466   b $L34
467 $L32:
468   lw  $v0,28($fp)
469   move  $sp,$fp
470   lw  $ra,40($sp)
471   lw  $fp,36($sp)
472   addu  $sp,$sp,48
473   j $ra
474   .end  read_word
475   .size read_word, .-read_word
476   .rdata
477   .align  2
478 $LC3:
479   .ascii  "help\000"
480   .align  2
481 $LC4:
482   .ascii  "version\000"
483   .align  2
484 $LC5:
485   .ascii  "input\000"
486   .align  2
487 $LC6:
488   .ascii  "output\000"
489   .data
490   .align  2
491   .type long_options.0, @object
492   .size long_options.0, 80
493 long_options.0:
```

```
494   .word $LC3
495   .word 0
496   .word 0
497   .word 104
498   .word $LC4
499   .word 0
500   .word 0
501   .word 86
502   .word $LC5
503   .word 1
504   .word 0
505   .word 105
506   .word $LC6
507   .word 1
508   .word 0
509   .word 111
510   .word 0
511   .word 0
512   .word 0
513   .word 0
514   .rdata
515   .align  2
516 $LC7:
517   .ascii  "hVui:o:\000"
518   .align  2
519 $LC8:
520   .ascii  "r\000"
521   .align  2
522 $LC9:
523   .ascii  "can't open input file, errno = %d\n\000"
524   .align  2
525 $LC10:
526   .ascii  "w\000"
527   .align  2
528 $LC11:
529   .ascii  "Can't open output file, errno = %d\n\000"
530   .align  2
531 $LC12:
532   .ascii  "%s\n\000"
533   .text
534   .align  2
535   .globl  main
536   .ent  main
537 main:
538   .frame  $fp,120,$ra   # vars= 72, regs= 3/0, args= 24, extra=
          8
539   .mask 0xd0000000,-8
540   .fmask  0x00000000,0
541   .set  noreorder
542   .cpload $t9
543   .set  reorder
544   subu  $sp,$sp,120
545   .cprestore 24
546   sw  $ra,112($sp)
```

```
547   sw  $fp,108($sp)
548   sw  $gp,104($sp)
549   move  $fp,$sp
550   sw  $a0,120($fp)
551   sw  $a1,124($fp)
552   sw  $zero,32($fp)
553   li  $v0,-1      # 0xffffffffffffffff
554   sw  $v0,36($fp)
555   li  $v0,-1      # 0xffffffffffffffff
556   sw  $v0,40($fp)
557   li  $v0,-1      # 0xffffffffffffffff
558   sw  $v0,44($fp)
559   li  $v0,-1      # 0xffffffffffffffff
560   sw  $v0,48($fp)
561   sw  $zero,52($fp)
562   sw  $zero,56($fp)
563   sw  $zero,60($fp)
564 $L44:
565   addu  $v0,$fp,60
566   sw  $v0,16($sp)
567   lw  $a0,120($fp)
568   lw  $a1,124($fp)
569   la  $a2,$LC7
570   la  $a3,long_options.0
571   la  $t9,getopt_long
572   jal $ra,$t9
573   sw  $v0,32($fp)
574   lw  $v1,32($fp)
575   li  $v0,-1      # 0xffffffffffffffff
576   bne $v1,$v0,$L46
577   b $L45
578 $L46:
579   lw  $v0,32($fp)
580   addu  $v0,$v0,-63
581   sw  $v0,96($fp)
582   lw  $v1,96($fp)
583   sltu  $v0,$v1,49
584   beq $v0,$zero,$L53
585   lw  $v0,96($fp)
586   sll $v1,$v0,2
587   la  $v0,$L54
588   addu  $v0,$v1,$v0
589   lw  $v0,0($v0)
590   .cpadd  $v0
591   j $v0
592   .rdata
593   .align  2
594 $L54:
595   .gpword $L52
596   .gpword $L53
597   .gpword $L53
598   .gpword $L53
599   .gpword $L53
600   .gpword $L53
```

```
601   .gpword $L53
602   .gpword $L53
603   .gpword $L53
604   .gpword $L53
605   .gpword $L53
606   .gpword $L53
607   .gpword $L53
608   .gpword $L53
609   .gpword $L53
610   .gpword $L53
611   .gpword $L53
612   .gpword $L53
613   .gpword $L53
614   .gpword $L53
615   .gpword $L53
616   .gpword $L53
617   .gpword $L53
618   .gpword $L49
619   .gpword $L53
620   .gpword $L53
621   .gpword $L53
622   .gpword $L53
623   .gpword $L53
624   .gpword $L53
625   .gpword $L53
626   .gpword $L53
627   .gpword $L53
628   .gpword $L53
629   .gpword $L53
630   .gpword $L53
631   .gpword $L53
632   .gpword $L53
633   .gpword $L53
634   .gpword $L53
635   .gpword $L53
636   .gpword $L48
637   .gpword $L50
638   .gpword $L53
639   .gpword $L53
640   .gpword $L53
641   .gpword $L53
642   .gpword $L53
643   .gpword $L51
644   .text
645 $L48:
646   sw  $zero,36($fp)
647   b $L44
648 $L49:
649   sw  $zero,40($fp)
650   b $L44
651 $L50:
652   sw  $zero,44($fp)
653   lw  $v0,optarg
654   sw  $v0,52($fp)
```

```
655   b $L44
656 $L51:
657   sw  $zero,48($fp)
658   lw  $v0,optarg
659   sw  $v0,56($fp)
660   b $L44
661 $L52:
662   li  $a0,1     # 0x1
663   la  $t9,exit
664   jal $ra,$t9
665 $L53:
666   la  $t9,print_usage
667   jal $ra,$t9
668   li  $a0,1     # 0x1
669   la  $t9,exit
670   jal $ra,$t9
671 $L45:
672   lw  $v0,36($fp)
673   bne $v0,$zero,$L55
674   la  $t9,print_help
675   jal $ra,$t9
676   move  $a0,$zero
677   la  $t9,exit
678   jal $ra,$t9
679 $L55:
680   lw  $v0,40($fp)
681   bne $v0,$zero,$L56
682   la  $t9,print_version
683   jal $ra,$t9
684   move  $a0,$zero
685   la  $t9,exit
686   jal $ra,$t9
687 $L56:
688   la  $v0,__sF
689   sw  $v0,64($fp)
690   la  $v0,__sF+88
691   sw  $v0,68($fp)
692   lw  $v0,44($fp)
693   bne $v0,$zero,$L58
694   lw  $a0,52($fp)
695   la  $a1,$LC8
696   la  $t9,fopen
697   jal $ra,$t9
698   sw  $v0,64($fp)
699   lw  $v0,64($fp)
700   bne $v0,$zero,$L58
701   la  $t9,__errno
702   jal $ra,$t9
703   la  $a0,$LC9
704   lw  $a1,0($v0)
705   la  $t9,printf
706   jal $ra,$t9
707   li  $v0,1     # 0x1
708   sw  $v0,92($fp)
```

```
709   b $L43
710 $L58:
711   lw  $v0,48($fp)
712   bne $v0,$zero,$L60
713   lw  $a0,56($fp)
714   la  $a1,$LC10
715   la  $t9,fopen
716   jal $ra,$t9
717   sw  $v0,68($fp)
718   lw  $v0,68($fp)
719   bne $v0,$zero,$L60
720   la  $t9,__errno
721   jal $ra,$t9
722   la  $a0,$LC11
723   lw  $a1,0($v0)
724   la  $t9,printf
725   jal $ra,$t9
726   li  $v1,1     # 0x1
727   sw  $v1,92($fp)
728   b $L43
729 $L60:
730   addu  $v0,$fp,72
731   move  $a0,$v0
732   lw  $a1,N
733   la  $t9,init_array
734   jal $ra,$t9
735   addu  $v0,$fp,72
736   lw  $a0,64($fp)
737   move  $a1,$v0
738   la  $t9,read_word
739   jal $ra,$t9
740   sw  $v0,88($fp)
741 $L62:
742   lw  $v1,88($fp)
743   li  $v0,1     # 0x1
744   beq $v1,$v0,$L64
745   b $L63
746 $L64:
747   addu  $v0,$fp,72
748   move  $a0,$v0
749   la  $t9,es_capicua
750   jal $ra,$t9
751   beq $v0,$zero,$L65
752   lw  $a0,68($fp)
753   la  $a1,$LC12
754   lw  $a2,72($fp)
755   la  $t9,fprintf
756   jal $ra,$t9
757 $L65:
758   addu  $v0,$fp,72
759   move  $a0,$v0
760   la  $t9,clear_array
761   jal $ra,$t9
762   addu  $v0,$fp,72
```

```
763   lw  $a0,64($fp)
764   move  $a1,$v0
765   la  $t9,read_word
766   jal $ra,$t9
767   sw  $v0,88($fp)
768   b $L62
769 $L63:
770   addu  $v0,$fp,72
771   move  $a0,$v0
772   la  $t9,free_array
773   jal $ra,$t9
774   lw  $v0,44($fp)
775   bne $v0,$zero,$L66
776   lw  $a0,64($fp)
777   la  $t9,fclose
778   jal $ra,$t9
779 $L66:
780   lw  $v0,48($fp)
781   bne $v0,$zero,$L67
782   lw  $a0,68($fp)
783   la  $t9,fclose
784   jal $ra,$t9
785 $L67:
786   sw  $zero,92($fp)
787 $L43:
788   lw  $v0,92($fp)
789   move  $sp,$fp
790   lw  $ra,112($sp)
791   lw  $fp,108($sp)
792   addu  $sp,$sp,120
793   j $ra
794   .end  main
795   .size main, .-main
796   .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

8. Enunciado

El enunciado se encuentra anexado al final de este documento.