Facultad de Ingeniería - U.B.A.

66.20 Organización de Computadoras - Práctica Martes

2do. Cuatrimestre de 2017

Trabajo práctico Nº 1

Programación MIPS

Matias Leandro Feld, Padrón: 99170
feldmatias@gmail.com

Federico Funes, Padrón: 98372
fede.funes96@gmail.com

Agustín Zorzano, Padrón: 99224
aguszorza@gmail.com

1. Documentación e implementación

El objetivo del trabajo es realizar un programa en lenguaje MIPS32 que lea palabras de una archivo (o de entrada estandar) y guarde en otro archivo (mostrar por salida estandar) únicamente aquellas palabras que sean palíndromos. Además, para analizar como influyen en el tiempo de ejecución las lecturas y escrituras en archivos, se implementó un sistema de buffer. Esto significa que al leer de un archivo no se hará de a un caracter por vez, sino que se llenará el buffer de entrada y luego se leerán los caracteres desde éste. Asimismo, para la escritura de archivos se realizará algo similar. Se guardarán en el buffer los caracteres a escribir, y se escribirán en el archivo una vez que el buffer se llene. De este modo, variando el tamaño del buffer, se podrá analizar como afectan al tiempo las operaciones con archivos.

El programa se divide en las siguientes funciones:

1. La función principal, main, que se encargara de la lógica de leer los parámetros de entrada y el manejo de los archivos. Si algun archivo no se puede abrir, no se pasaron correctamente los parámetros el programa, o se produjo un error en la ejecución, mostrará un mensaje de error en el archivo stderr y finalizará con un código de error. Esta funcion será escrita en lenguaje C.

2. La función palindrome, que es la que se encarga del bucle principal. que consiste en leer una palabra del archivo de entrada, comprobar si es palíndromo y escribirla en el archivo de salida si corresponde. Ésta es la función de entrada al programa en MIPS que deberá ser llamada desde el programa en C. Recibe por parámetro el archivo de entrada, el de salida y los tamaños de los buffer. Al ser llamada lo primero que hará es crear los buffer de entrada y salida, utilizando la función crear_buffer(). Luego entrará en el bucle hasta que todos los caracteres del archivo de entrada sean analizados. El bucle termina cuando se lee el EOF, y en este caso se llamará una vez más a la funciónque escribe en archivos para escribir todo lo que haya quedado en el buffer de salida. El stackframe correspondiente a esta función quedará definido de la siguiente manera:

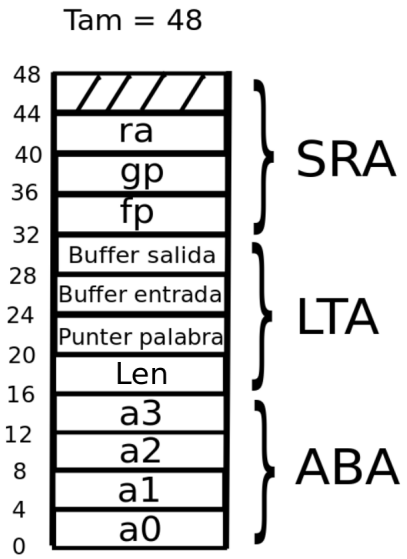

Figura 1: Stackframe de leer_archivo

3. La función leer palabra, que se encarga de leer una palabra del archivo. Debido a las limitaciones de lo que se considera palabra, y a que no hay limitación con respecto a cantidad de letras de una palabra, lo que hacemos es leer carácter por carácter, guardándolos en

un vector alojado en memoria dinámica que se irá redimensionando a medida que sea necesario. Para ello, definimos una variable TAM que determinará la cantidad de memoria que se pide al inicio y al redimensionar. En principio esa variable puede contener cualquier número, pero para no estar redimensionando muchas veces y para no pedir mucha memoria innecesaria, definimos ese valor en 50. La función recibe por parámetro un puntero a entero, que sirve para guardar la longitud de la palabra leída, con el objetivo de no tener que calcularla nuevamente en otro momento. Para leer un caracter llamará a la función getch(). Para facilitar la escritura de la palabra, al final de la misma se insertará un \n en lugar de un \0. El stackframe correspondiente a la función quedará definido de la siguiente manera:

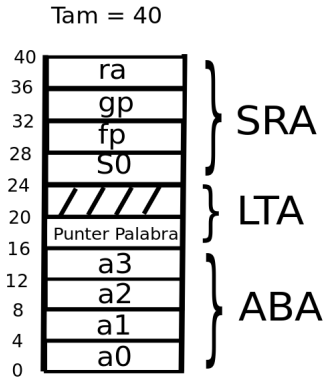

Figura 2: Stackframe de leer_palabra

4. La función es capicúa, que se encarga de comprobar si la palabra es o no un palíndromo, y devuelve un valor booleano según corresponda. Ésta función recibe por parámetro el puntero a la palabra y la longitud de la misma. El stackframe correspondiente a esta función quedará definido de la siguiente manera:

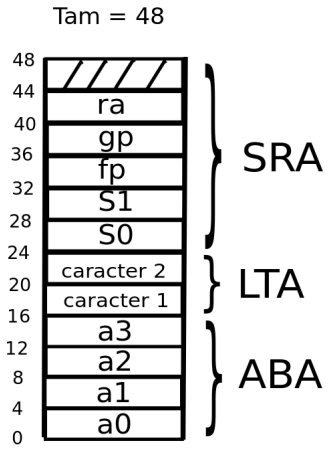

Figura 3: Stackframe de es_capicua

5. La función my_tolower, que fue implementada para reemplazar la del lenguaje C, se encarga de pasar a minúscula un caracter. Para eso, recibe por parámetro el caracter, y lo transforma únicamente si es una letra mayúscula, caso contrario lo devuelve como viene. El stackframe correspondiente a esta función quedará definido de la siguiente manera:

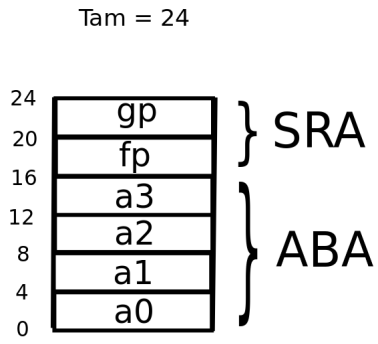

Figura 4: Stackframe de my_tolower

6. La función crear buffer, es la encargada de crear los buffers. Para ello recibirá por parámetro el tamaño del mismo, y lo creará haciendo uso de la función malloc. El stackframe correspondiente a esta función quedará definido de la siguiente manera:

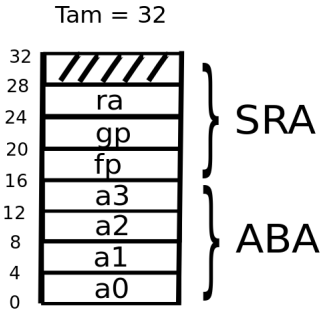

Figura 5: Stackframe de crear_buffer

7. La función getch, que se encarga de leer un carácter del archivo de entrada. Como se explicó anteriormente, ésta hace uso de un buffer. Por lo tanto, conociendo el tamaño del buffer y la última posición leída, devolverá el caracter correspondiente, y cuando la posición sea mayor o igual al tamaño se encargará de llenar el buffer nuevamente con nuevos datos. Esta función tiene una complicación adicional, ya que debe indicar cuando fue leído el final del archivo en el buffer. Para eso, utilizaremos una variable global, que será nula hasta el momento en que se lee el EOF, que cambiará de valor y permitirá avisar a las demás funciones que ya se leyó todo el archivo. Si se produjera algún error en la lectura devolverá un código de error. Para la lectura del archivo hace uso de un syscall. Puede ocurrir que se lean menos bytes de los pedidos, en ese caso pueden ser por dos razones, que no hay más por leer o que se leyó menos pero se puede leer más. Esto lo solucionamos haciendo que la lectura se haga en un loop, que termina cuando no hay más para leer o cuando se llenó el buffer. El stackframe correspondiente a esta función quedará definido de la siguiente manera:

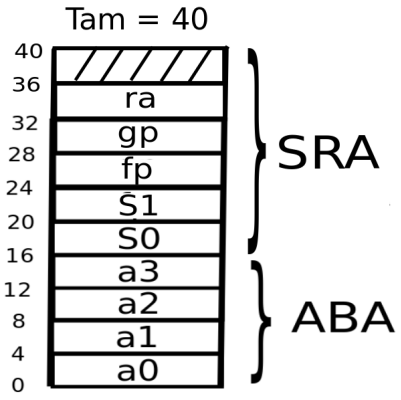

Figura 6: Stackframe de getch

8. La función putch, que se encarga de escribir una palabra en el archivo de salida. Debido a que debe utilizar el buffer, la función recibirá por parámetro la palabra, y guardará de a un caracter por vez en el buffer. Una vez que se llene el buffer, independientemente si se guardó toda la palabra o no, éste se escribirá en el archivo y se vaciará. Al igual que la anterior, también tiene una complicación. Puede ocurrir que el buffer no se llene completamente y se haya terminado el archivo, en cuyo caso, utilizando la variable global que indica si se debe escribir el EOF, escribirá todo lo que se encuentre en el buffer en ese momento. El stackframe correspondiente a esta función quedará definido de la siguiente manera:

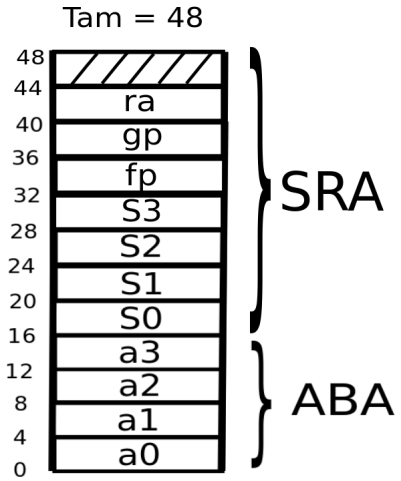

Figura 7: Stackframe de putch

9. Por último, la función myrealloc, que sirve para redimensionar un bloque de memoria dinámica. Para facilitar la programación de la misma, se decidió que solo se podrá redimensionar aumentando el tamaño del bloque y no disminuyéndolo. Por eso, la función recibe por parámetro el puntero al bloque, el tamaño actual, y el tamaño a agregar. Haciendo uso de la función mymalloc crea un nuevo bloque y copia byte por byte los datos del bloque viejo al nuevo. Finalmente libera el bloque viejo y devuelve el nuevo. Si se produjera un error al llamar a la función mymalloc se devolverá un puntero a NULL. El stackframe correspondiente a esta función quedará definido de la siguiente manera:

Figura 8: Stackframe de myrealloc

2. Comandos para compilacion

Para compilar el programa utilizamos el siguiente comando:

$ gcc -Wall -o tp1 main.c mymalloc.S myrealloc.S palindrome.S

o se puede optar por ejecutar el script de bash:

$ ./compilar.sh

3. Pruebas

Para probar el programa utilizamos un script de bash llamado ’pruebas.sh” que contiene un conjunto de pruebas que se realizan automáticamente. Entre ellas,se encuentran pruebas con archivos vacios, archivos con un solo caracter y archivos solo con simbolos. Por otro lado, tambien se prueba que funcionen correctamente los mensajes de error cuando los parámetros no son usados correctamente. Se realizan pruebas para distintos tamaños de buffer para asegurarnos que funcione correctamente. Todas las pruebas utilizan el siguiente comando:

$ diff salida.txt resultado.txt

Donde si no muestra nada significa que ambos archivos son iguales, y que por lo tanto todas las pruebas del programa funcionan correctamente.

En una de las pruebas utilizamos un archivo de texto ”entrada.txt” que contiene un conjunto de palabras con combinaciones de letras, numeros y guiones y mezclando mayúsculas y minúsculas. Luego tenemos otro archivo, ”resultado.txt” que es lo que se espera que devuelva el programa al ejecutarse con ese archivo de entrada. En la siguiente sección se muestran esos archivos. En el resto de las pruebas se usan archivos creados dentro del mismo script, que se borran al finalizar.

También realizamos pruebas utilizando salida estándar y entrada estándar, los cuales funcionaron correctamente. Cuando se trabaja con entrada estándar y se desea finalizar se debe ingresar ”ctrl D”, que inserta un EOF, ya que utilizando ”ctrl C” finaliza abruptamente y no se guarda correctamente el resultado.

3.1. Archivo 'pruebas.sh'

```
#/bin/bash

gcc -Wall -o tp0 tp0.c


# Prueba con archivo de pruebas
./tp0 -i entrada.txt -o salida.txt
diff salida.txt resultado.txt

# Prueba con archivo vacio
touch vacio.txt
touch resultado_vacio.txt
./tp0 -i vacio.txt -o salida.txt
diff salida.txt resultado_vacio.txt

# Prueba con una sola letra mayúscula
echo M | ./tp0 -o salida.txt
echo M > res.txt
diff salida.txt res.txt

# Prueba con una sola letra minúscula
echo m | ./tp0 -o salida.txt
echo m > res.txt
diff salida.txt res.txt

# Prueba con un número
echo 3 | ./tp0 -o salida.txt
echo 3 > res.txt
diff salida.txt res.txt

# Prueba con un guion
echo - | ./tp0 -o salida.txt
echo - > res.txt
diff salida.txt res.txt

# Prueba con un guion bajo
echo _ | ./tp0 -o salida.txt
echo _ > res.txt
diff salida.txt res.txt

# Prueba con un simbolo
echo @ | ./tp0 -o salida.txt
diff salida.txt vacio.txt

# Prueba con espacios
echo "                    " > ent.txt
./tp0 -i ent.txt -o salida.txt
diff salida.txt vacio.txt
```

```
# Prueba con simbolos
echo "@#$%^*()!{}[],./?<>;:*+\|=+" > ent.txt
./tp0 -i ent.txt -o salida.txt
diff salida.txt vacio.txt

# Prueba error: no se ingresa archivo de entrada
echo "Debe indicar un archivo de entrada luego de -i" > res.txt
./tp0 -i 2> error.txt
diff error.txt res.txt

# Prueba error: no se ingresa archivo de entrada
echo "Debe indicar un archivo de entrada luego de -i" > res.txt
./tp0 -o salida.txt -i 2> error.txt
diff error.txt res.txt

# Prueba error: no se ingresa archivo de salida
echo "Debe indicar un archivo de salida luego de -o" > res.txt
./tp0 -o 2> error.txt
diff error.txt res.txt

# Prueba error: no se ingresa archivo de salida
echo "Debe indicar un archivo de salida luego de -o" > res.txt
./tp0 -i entrada.txt -o 2> error.txt
diff error.txt res.txt

# Prueba error: no se puede abrir el archivo de entrada
echo "El archivo de entrada no pudo abrirse" > res.txt
./tp0 -i inexistente.txt 2> error.txt
diff error.txt res.txt

#Prueba con stdin
./tp0 -o salida.txt  < entrada.txt
diff salida.txt resultado.txt

#Prueba con stdin
./tp0 -i - -o salida.txt < entrada.txt
diff salida.txt resultado.txt

#Prueba con stdout
./tp0 -i entrada.txt > salida.txt
diff salida.txt resultado.txt

#Prueba con stdout
./tp0 -i entrada.txt -o - > salida.txt
diff salida.txt resultado.txt

#Borramos archivos sobrantes
rm vacio.txt
rm resultado_vacio.txt
```

```
rm salida.txt
rm ent.txt
rm error.txt
rm res.txt
```

3.2. Archivo 'entrada.txt'

```
Pruebas varias:
aaa      pelota hola como estas

pepep aaaaaaaaaaaaaaa aaaaaaaaaaaaaaaa
_aa_

_aAAa_
-a-a-
-a-a
Neuquen
-Neuquen-    neu%q%uen

1234321   ?123?123abc4cba321

Prueba del enunciado:

Somos los primeros en completar el TP 0.
Ojo que La fecha de entrega del TP0 es el martes 12 de septiembre.

Palabras largas mezcladas:

abcdefghijklmnopqrstuvwxyz0123456789_--_9876543210zyxwvutsrqponmlkjihgfedcba??==
ABCDEFGHIJKLMnopqrstuvwxyz0123456789_--_9876543210zyxwvutsrqponmlkjihgfedcba??==
EstoesUnPalindromoOMOrdnilapNUSEOTse.........EStono

Pruebas de guiones  guiones bajos:

__--__???????######$$$$_-_-_@@@@@-_-__-_-!    ---___

Pruebas de palabras de una letra:

a    %%%%1 2 ^4^ - _ C D
b  ! @ # $ %^ & * ( ) = + \
c
d

Pruebas solo mayusculas:

AAA ABCDEDCBA    ABC123--321CBA   WXXW

PALINDROMO   -ABCB-
```

3.3. Archivo 'resultado.txt'

```
aaa
pepep
aaaaaaaaaaaaaaa
```

```
aaaaaaaaaaaaaaaa
_aa_
_aAAa_
-a-a-
Neuquen
-Neuquen-
q
1234321
123abc4cba321
Somos
0
Ojo
abcdefghijklmnopqrstuvwxyz0123456789_--_9876543210zyxwvutsrqponmlkjihgfedcba
ABCDEFGHIJKLMnopqrstuvwxyz0123456789_--_9876543210zyxwvutsrqponmlkjihgfedcba
EstoesUnPalindromoOMOrdnilapNUSEOTse
__--__
_-_-_
-_-__-_-
a
1
2
4
-

_
C
D
b
c
d
AAA
ABCDEDCBA
ABC123--321CBA
WXXW
```

4. Código fuente

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdbool.h>
4  #include <stdlib.h>
5  #include <ctype.h>
6  #define TAM 10
7
8
9  char * leer_palabra(FILE* archivo, int* longitud){
10         char* palabra = realloc(NULL,TAM);
11         int len = 0;
12         while(true){
13                 int c = fgetc(archivo);
14                 if (ferror(archivo)){
15                         free(palabra);
16                         return NULL;
```

```
17                  }
18                  if((c>='A' && c<='Z') ||(c>='a' && c<='z') || (c>='0'
                        && c<='9') || (c == '-') || (c == '_')){
19                          palabra[len] = c;
20                          len ++;
21                          if (len %TAM == 0){
22                                  palabra = realloc(palabra, TAM + len)
                                        ;
23                          }
24                  }
25                  else{
26                          palabra[len] = '\0';
27                          *longitud = len;
28                          return palabra;
29                  }
30          }
31  }
32
33  bool es_capicua(char* palabra, int len){
34          if (len == 0){
35                  return false;
36          }
37          int inicio = 0;
38          int final = len - 1;
39          while(inicio < final){
40                  if (tolower((unsigned char)palabra[inicio]) !=
                        tolower((unsigned char)palabra[final])){
41                          return false;
42                  }
43                  inicio++;
44                  final--;
45          }
46          return true;
47  }
48
49  int main(int argc, char* argv[]){
50          FILE* entrada = stdin;
51          FILE* salida = stdout;
52          char* parametro;
53
54          int i;
55          for (i = 1; i < argc; i += 2){
56                  if (strcmp(argv[i],"-i") == 0){
57                          if (i + 1 >= argc){
58                                  fputs("Debe␣indicar␣un␣archivo␣de␣
                                        entrada␣luego␣de␣-i\n", stderr);
59                                  return 2;
60                          }
61                          parametro = argv[i + 1];
62                          if (strcmp(parametro,"-") != 0){
```

```
63                                      entrada = fopen(argv[i + 1], "r");
64                                      if (!entrada){
65                                              fputs("El archivo de entrada
                                                   no pudo abrirse\n", stderr
                                                   );
66                                              return 1;
67                                      }
68                              }
69                      }
70                      else if (strcmp(argv[i],"-o") == 0){
71                              if (i + 1 >= argc){
72                                      fputs("Debe indicar un archivo de
                                           salida luego de -o\n", stderr);
73                                      return 2;
74                              }
75                              parametro = argv[i + 1];
76                              if (strcmp(parametro,"-") != 0){
77                                      salida = fopen(argv[i + 1], "w");
78                                      if (!salida){
79                                              fputs("El archivo de salida
                                                   no pudo abrirse\n", stderr
                                                   );
80                                              return 1;
81                                      }
82                              }
83                      }
84                      else if (strcmp(argv[i],"-V") == 0){
85                              fprintf(stdout, "TP0 version 1.0002\n");
86                              return 0;
87                      }
88                      else if (strcmp(argv[i],"-h") == 0){
89                              fprintf(stdout, "Usage:\n\ntp0 -h\ntp0 -V\
                                   ntp0 [options]\n\nOptions:\n-V, --version
                                   Print version and quit.\n-h, --help
                                   Print this information.\n-i, --input
                                   Location of the input file.\n-o, --output
                                   Location of the output file.\n\nExample
                                   :\ntp0 -i ~/input -o ~/output\n");
90                              return 0;
91                      }
92              }
93
94              char* palabra;
95              int len;
96              while(!feof(entrada)){
97                      palabra = leer_palabra(entrada, &len);
98                      if (!palabra){
99                              fputs("Ocurrio un error inesperado\n", stderr
                                   );
100                             return 3;
```

```
101                 }
102                 if (es_capicua(palabra, len)){
103                         fprintf(salida, "%s\n", palabra);
104                 }
105                 free (palabra);
106         }
107
108         fclose(entrada);
109         fclose(salida);
110
111         return 0;
112 }
```

5. Codigo MIPS32

```
  1         .file   1 "tp0.c"
  2         .section .mdebug.abi32
  3         .previous
  4         .abicalls
  5         .text
  6         .align  2
  7         .globl  leer_palabra
  8         .ent    leer_palabra
  9 leer_palabra:
 10         .frame  $fp,56,$ra              # vars= 16, regs= 3/0, args= 16,
              extra= 8
 11         .mask   0xd0000000,-8
 12         .fmask  0x00000000,0
 13         .set    noreorder
 14         .cpload $t9
 15         .set    reorder
 16         subu    $sp,$sp,56
 17         .cprestore 16
 18         sw      $ra,48($sp)
 19         sw      $fp,44($sp)
 20         sw      $gp,40($sp)
 21         move    $fp,$sp
 22         sw      $a0,56($fp)
 23         sw      $a1,60($fp)
 24         move    $a0,$zero
 25         li      $a1,10                  # 0xa
 26         la      $t9,realloc
 27         jal     $ra,$t9
 28         sw      $v0,24($fp)
 29         sw      $zero,28($fp)
 30 $L18:
 31         lw      $a0,56($fp)
 32         la      $t9,fgetc
 33         jal     $ra,$t9
 34         sw      $v0,32($fp)
 35         lw      $v0,56($fp)
 36         lhu     $v0,12($v0)
```

```
37          srl     $v0,$v0,6
38          andi    $v0,$v0,0x1
39          beq     $v0,$zero,$L21
40          lw      $a0,24($fp)
41          la      $t9,free
42          jal     $ra,$t9
43          sw      $zero,36($fp)
44          b       $L17
45  $L21:
46          lw      $v0,32($fp)
47          slt     $v0,$v0,65
48          bne     $v0,$zero,$L24
49          lw      $v0,32($fp)
50          slt     $v0,$v0,91
51          bne     $v0,$zero,$L23
52  $L24:
53          lw      $v0,32($fp)
54          slt     $v0,$v0,97
55          bne     $v0,$zero,$L25
56          lw      $v0,32($fp)
57          slt     $v0,$v0,123
58          bne     $v0,$zero,$L23
59  $L25:
60          lw      $v0,32($fp)
61          slt     $v0,$v0,48
62          bne     $v0,$zero,$L26
63          lw      $v0,32($fp)
64          slt     $v0,$v0,58
65          bne     $v0,$zero,$L23
66  $L26:
67          lw      $v1,32($fp)
68          li      $v0,45                  # 0x2d
69          beq     $v1,$v0,$L23
70          lw      $v1,32($fp)
71          li      $v0,95                  # 0x5f
72          beq     $v1,$v0,$L23
73          b       $L22
74  $L23:
75          lw      $v1,24($fp)
76          lw      $v0,28($fp)
77          addu    $v1,$v1,$v0
78          lbu     $v0,32($fp)
79          sb      $v0,0($v1)
80          lw      $v0,28($fp)
81          addu    $v0,$v0,1
82          sw      $v0,28($fp)
83          lw      $a0,28($fp)
84          li      $v0,1717960704                  # 0x66660000
85          ori     $v0,$v0,0x6667
86          mult    $a0,$v0
87          mfhi    $v0
88          sra     $v1,$v0,2
89          sra     $v0,$a0,31
```

```
 90          subu    $v1,$v1,$v0
 91          move    $v0,$v1
 92          sll     $v0,$v0,2
 93          addu    $v0,$v0,$v1
 94          sll     $v0,$v0,1
 95          subu    $v0,$a0,$v0
 96          bne     $v0,$zero,$L18
 97          lw      $v0,28($fp)
 98          addu    $v0,$v0,10
 99          lw      $a0,24($fp)
100          move    $a1,$v0
101          la      $t9,realloc
102          jal     $ra,$t9
103          sw      $v0,24($fp)
104          b       $L18
105  $L22:
106          lw      $v1,24($fp)
107          lw      $v0,28($fp)
108          addu    $v0,$v1,$v0
109          sb      $zero,0($v0)
110          lw      $v1,60($fp)
111          lw      $v0,28($fp)
112          sw      $v0,0($v1)
113          lw      $v0,24($fp)
114          sw      $v0,36($fp)
115  $L17:
116          lw      $v0,36($fp)
117          move    $sp,$fp
118          lw      $ra,48($sp)
119          lw      $fp,44($sp)
120          addu    $sp,$sp,56
121          j       $ra
122          .end    leer_palabra
123          .size   leer_palabra, .-leer_palabra
124          .align  2
125          .globl  es_capicua
126          .ent    es_capicua
127  es_capicua:
128          .frame  $fp,32,$ra              # vars= 16, regs= 2/0, args= 0,
                extra= 8
129          .mask   0x50000000,-4
130          .fmask  0x00000000,0
131          .set    noreorder
132          .cpload $t9
133          .set    reorder
134          subu    $sp,$sp,32
135          .cprestore 0
136          sw      $fp,28($sp)
137          sw      $gp,24($sp)
138          move    $fp,$sp
139          sw      $a0,32($fp)
140          sw      $a1,36($fp)
141          lw      $v0,36($fp)
```

```
142         bne     $v0,$zero,$L30
143         sw      $zero,16($fp)
144         b       $L29
145 $L30:
146         sw      $zero,8($fp)
147         lw      $v0,36($fp)
148         addu    $v0,$v0,-1
149         sw      $v0,12($fp)
150 $L31:
151         lw      $v0,8($fp)
152         lw      $v1,12($fp)
153         slt     $v0,$v0,$v1
154         bne     $v0,$zero,$L33
155         b       $L32
156 $L33:
157         lw      $v1,32($fp)
158         lw      $v0,8($fp)
159         addu    $v0,$v1,$v0
160         lbu     $v0,0($v0)
161         sll     $v1,$v0,1
162         lw      $v0,_tolower_tab_
163         addu    $v0,$v1,$v0
164         addu    $a0,$v0,2
165         lw      $v1,32($fp)
166         lw      $v0,12($fp)
167         addu    $v0,$v1,$v0
168         lbu     $v0,0($v0)
169         sll     $v1,$v0,1
170         lw      $v0,_tolower_tab_
171         addu    $v0,$v1,$v0
172         addu    $v0,$v0,2
173         lh      $v1,0($a0)
174         lh      $v0,0($v0)
175         beq     $v1,$v0,$L34
176         sw      $zero,16($fp)
177         b       $L29
178 $L34:
179         lw      $v0,8($fp)
180         addu    $v0,$v0,1
181         sw      $v0,8($fp)
182         lw      $v0,12($fp)
183         addu    $v0,$v0,-1
184         sw      $v0,12($fp)
185         b       $L31
186 $L32:
187         li      $v0,1                   # 0x1
188         sw      $v0,16($fp)
189 $L29:
190         lw      $v0,16($fp)
191         move    $sp,$fp
192         lw      $fp,28($sp)
193         addu    $sp,$sp,32
194         j       $ra
```

```
195         .end    es_capicua
196         .size   es_capicua, .-es_capicua
197         .rdata
198         .align  2
199 $LC0:
200         .ascii  "-i\000"
201         .align  2
202 $LC1:
203         .ascii  "Debe indicar un archivo de entrada luego de -i\n\000"
204         .align  2
205 $LC2:
206         .ascii  "-\000"
207         .align  2
208 $LC3:
209         .ascii  "r\000"
210         .align  2
211 $LC4:
212         .ascii  "El archivo de entrada no pudo abrirse\n\000"
213         .align  2
214 $LC5:
215         .ascii  "-o\000"
216         .align  2
217 $LC6:
218         .ascii  "Debe indicar un archivo de salida luego de -o\n\000"
219         .align  2
220 $LC7:
221         .ascii  "w\000"
222         .align  2
223 $LC8:
224         .ascii  "El archivo de salida no pudo abrirse\n\000"
225         .align  2
226 $LC9:
227         .ascii  "-V\000"
228         .align  2
229 $LC10:
230         .ascii  "TP0 version 1.0002\n\000"
231         .align  2
232 $LC11:
233         .ascii  "-h\000"
234         .align  2
235 $LC12:
236         .ascii  "Usage:\n\n"
237         .ascii  "tp0 -h\n"
238         .ascii  "tp0 -V\n"
239         .ascii  "tp0 [options]\n\n"
240         .ascii  "Options:\n"
241         .ascii  "-V, --version  Print version and quit.\n"
242         .ascii  "-h, --help   Print this information.\n"
243         .ascii  "-i, --input   Location of the input file.\n"
244         .ascii  "-o, --output   Location of the output file.\n\n"
245         .ascii  "Example:\n"
246         .ascii  "tp0 -i ~/input -o ~/output\n\000"
247         .align  2
```

```
248 $LC13:
249         .ascii  "Ocurrio un error inesperado\n\000"
250         .align  2
251 $LC14:
252         .ascii  "%s\n\000"
253         .text
254         .align  2
255         .globl  main
256         .ent    main
257 main:
258         .frame  $fp,72,$ra              # vars= 32, 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,72
265         .cprestore 16
266         sw      $ra,64($sp)
267         sw      $fp,60($sp)
268         sw      $gp,56($sp)
269         move    $fp,$sp
270         sw      $a0,72($fp)
271         sw      $a1,76($fp)
272         la      $v0,__sF
273         sw      $v0,24($fp)
274         la      $v0,__sF+88
275         sw      $v0,28($fp)
276         li      $v0,1                   # 0x1
277         sw      $v0,36($fp)
278 $L36:
279         lw      $v0,36($fp)
280         lw      $v1,72($fp)
281         slt     $v0,$v0,$v1
282         bne     $v0,$zero,$L39
283         b       $L37
284 $L39:
285         lw      $v0,36($fp)
286         sll     $v1,$v0,2
287         lw      $v0,76($fp)
288         addu    $v0,$v1,$v0
289         lw      $a0,0($v0)
290         la      $a1,$LC0
291         la      $t9,strcmp
292         jal     $ra,$t9
293         bne     $v0,$zero,$L40
294         lw      $v0,36($fp)
295         addu    $v1,$v0,1
296         lw      $v0,72($fp)
297         slt     $v0,$v1,$v0
298         bne     $v0,$zero,$L41
299         la      $a0,$LC1
```

```
300             la      $a1,__sF+176
301             la      $t9,fputs
302             jal     $ra,$t9
303             li      $v0,2                   # 0x2
304             sw      $v0,48($fp)
305             b       $L35
306     $L41:
307             lw      $v0,36($fp)
308             sll     $v1,$v0,2
309             lw      $v0,76($fp)
310             addu    $v0,$v1,$v0
311             addu    $v0,$v0,4
312             lw      $v0,0($v0)
313             sw      $v0,32($fp)
314             lw      $a0,32($fp)
315             la      $a1,$LC2
316             la      $t9,strcmp
317             jal     $ra,$t9
318             beq     $v0,$zero,$L38
319             lw      $v0,36($fp)
320             sll     $v1,$v0,2
321             lw      $v0,76($fp)
322             addu    $v0,$v1,$v0
323             addu    $v0,$v0,4
324             lw      $a0,0($v0)
325             la      $a1,$LC3
326             la      $t9,fopen
327             jal     $ra,$t9
328             sw      $v0,24($fp)
329             lw      $v0,24($fp)
330             bne     $v0,$zero,$L38
331             la      $a0,$LC4
332             la      $a1,__sF+176
333             la      $t9,fputs
334             jal     $ra,$t9
335             li      $v0,1                   # 0x1
336             sw      $v0,48($fp)
337             b       $L35
338     $L40:
339             lw      $v0,36($fp)
340             sll     $v1,$v0,2
341             lw      $v0,76($fp)
342             addu    $v0,$v1,$v0
343             lw      $a0,0($v0)
344             la      $a1,$LC5
345             la      $t9,strcmp
346             jal     $ra,$t9
347             bne     $v0,$zero,$L45
348             lw      $v0,36($fp)
349             addu    $v1,$v0,1
350             lw      $v0,72($fp)
351             slt     $v0,$v1,$v0
352             bne     $v0,$zero,$L46
```

```
353             la      $a0,$LC6
354             la      $a1,__sF+176
355             la      $t9,fputs
356             jal     $ra,$t9
357             li      $v0,2                   # 0x2
358             sw      $v0,48($fp)
359             b       $L35
360     $L46:
361             lw      $v0,36($fp)
362             sll     $v1,$v0,2
363             lw      $v0,76($fp)
364             addu    $v0,$v1,$v0
365             addu    $v0,$v0,4
366             lw      $v0,0($v0)
367             sw      $v0,32($fp)
368             lw      $a0,32($fp)
369             la      $a1,$LC2
370             la      $t9,strcmp
371             jal     $ra,$t9
372             beq     $v0,$zero,$L38
373             lw      $v0,36($fp)
374             sll     $v1,$v0,2
375             lw      $v0,76($fp)
376             addu    $v0,$v1,$v0
377             addu    $v0,$v0,4
378             lw      $a0,0($v0)
379             la      $a1,$LC7
380             la      $t9,fopen
381             jal     $ra,$t9
382             sw      $v0,28($fp)
383             lw      $v0,28($fp)
384             bne     $v0,$zero,$L38
385             la      $a0,$LC8
386             la      $a1,__sF+176
387             la      $t9,fputs
388             jal     $ra,$t9
389             li      $v0,1                   # 0x1
390             sw      $v0,48($fp)
391             b       $L35
392     $L45:
393             lw      $v0,36($fp)
394             sll     $v1,$v0,2
395             lw      $v0,76($fp)
396             addu    $v0,$v1,$v0
397             lw      $a0,0($v0)
398             la      $a1,$LC9
399             la      $t9,strcmp
400             jal     $ra,$t9
401             bne     $v0,$zero,$L50
402             la      $a0,__sF+88
403             la      $a1,$LC10
404             la      $t9,fprintf
405             jal     $ra,$t9
```

```
406         sw      $zero,48($fp)
407         b       $L35
408 $L50:
409         lw      $v0,36($fp)
410         sll     $v1,$v0,2
411         lw      $v0,76($fp)
412         addu    $v0,$v1,$v0
413         lw      $a0,0($v0)
414         la      $a1,$LC11
415         la      $t9,strcmp
416         jal     $ra,$t9
417         bne     $v0,$zero,$L38
418         la      $a0,__sF+88
419         la      $a1,$LC12
420         la      $t9,fprintf
421         jal     $ra,$t9
422         sw      $zero,48($fp)
423         b       $L35
424 $L38:
425         lw      $v0,36($fp)
426         addu    $v0,$v0,2
427         sw      $v0,36($fp)
428         b       $L36
429 $L37:
430         .set    noreorder
431         nop
432         .set    reorder
433 $L53:
434         lw      $v0,24($fp)
435         lhu     $v0,12($v0)
436         srl     $v0,$v0,5
437         andi    $v0,$v0,0x1
438         beq     $v0,$zero,$L55
439         b       $L54
440 $L55:
441         addu    $v0,$fp,44
442         lw      $a0,24($fp)
443         move    $a1,$v0
444         la      $t9,leer_palabra
445         jal     $ra,$t9
446         sw      $v0,40($fp)
447         lw      $v0,40($fp)
448         bne     $v0,$zero,$L56
449         la      $a0,$LC13
450         la      $a1,__sF+176
451         la      $t9,fputs
452         jal     $ra,$t9
453         li      $v0,3                   # 0x3
454         sw      $v0,48($fp)
455         b       $L35
456 $L56:
457         lw      $a0,40($fp)
458         lw      $a1,44($fp)
```

```
459         la      $t9,es_capicua
460         jal     $ra,$t9
461         beq     $v0,$zero,$L57
462         lw      $a0,28($fp)
463         la      $a1,$LC14
464         lw      $a2,40($fp)
465         la      $t9,fprintf
466         jal     $ra,$t9
467 $L57:
468         lw      $a0,40($fp)
469         la      $t9,free
470         jal     $ra,$t9
471         b       $L53
472 $L54:
473         lw      $a0,24($fp)
474         la      $t9,fclose
475         jal     $ra,$t9
476         lw      $a0,28($fp)
477         la      $t9,fclose
478         jal     $ra,$t9
479         sw      $zero,48($fp)
480 $L35:
481         lw      $v0,48($fp)
482         move    $sp,$fp
483         lw      $ra,64($sp)
484         lw      $fp,60($sp)
485         addu    $sp,$sp,72
486         j       $ra
487         .end    main
488         .size   main, .-main
489         .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```