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:

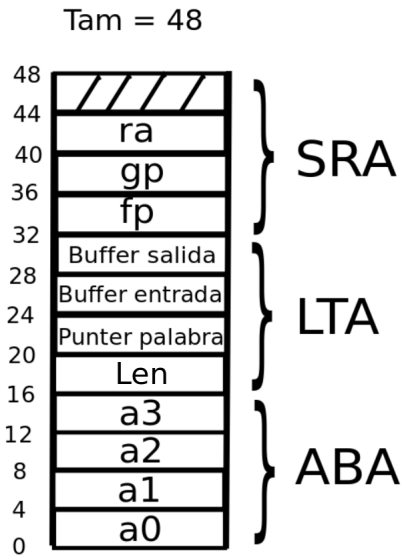

Figura 1: Stackframe de leer_archivo

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:

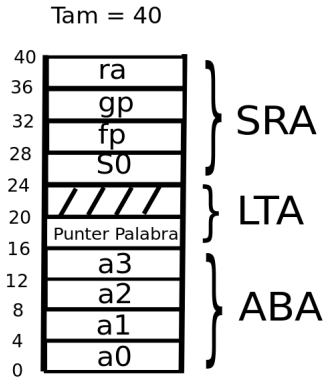

Figura 2: Stackframe de leer_palabra

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:

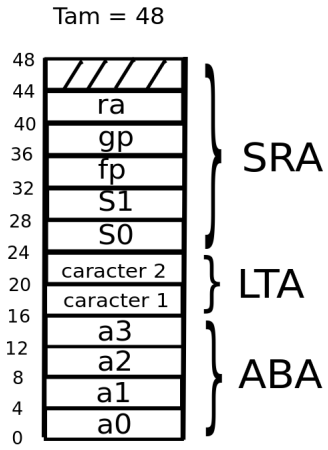

Figura 3: Stackframe de es_capicua

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:

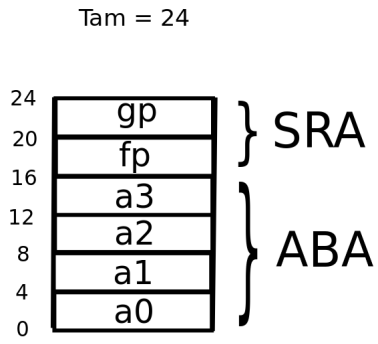

Figura 4: Stackframe de my_tolower

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:

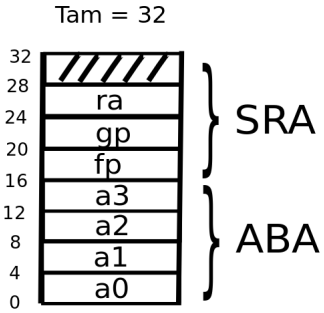

Figura 5: Stackframe de crear_buffer

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:

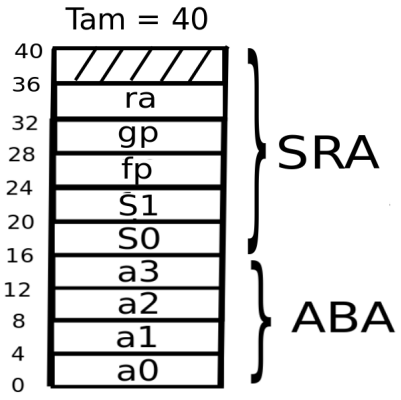

Figura 6: Stackframe de getch

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:

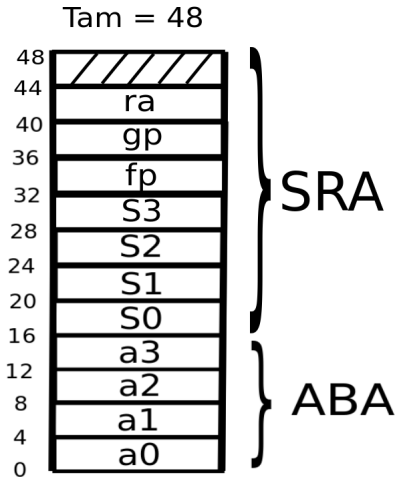

Figura 7: Stackframe de putch

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

./compilar.sh


if [ -f time.txt ]; then
    rm time.txt
fi

# Pruebas con archivo de pruebas entrada.txt y resultado.txt
echo -e "PRUEBAS CON ARCHIVO DE PRUEBAS ENTRADA.TXT Y RESULTADO.TXT\n" >>
   time.txt

echo -e "Resultado con 1 byte entrada y 1 byte salida:" >> time.txt

(time ./tp1 -i entrada.txt -o salida.txt -I 1 -O 1)2>>time.txt
diff salida.txt resultado.txt

echo -e "\nResultado con 20 bytes entrada y 20 bytes salida:" >> time.txt

(time ./tp1 -i entrada.txt -o salida.txt -I 20 -O 20)2>>time.txt
diff salida.txt resultado.txt

echo -e "\nResultado con 100 bytes entrada y 100 bytes salida:" >> time.txt
(time ./tp1 -i entrada.txt -o salida.txt -I 100 -O 100)2>>time.txt
diff salida.txt resultado.txt

echo -e "\nResultado con 1000 bytes entrada y 1000 bytes salida:" >> time.
   txt
(time ./tp1 -i entrada.txt -o salida.txt -I 1000 -O 1000)2>>time.txt
diff salida.txt resultado.txt

echo -e "\nResultado con 20 bytes entrada y 100 bytes salida:" >> time.txt
(time ./tp1 -i entrada.txt -o salida.txt -I 20 -O 100)2>>time.txt
diff salida.txt resultado.txt

echo -e "\nResultado con 100 bytes entrada y 20 bytes salida:" >> time.txt
(time ./tp1 -i entrada.txt -o salida.txt -I 100 -O 20)2>>time.txt
diff salida.txt resultado.txt

echo -e "\nResultado con 1 byte entrada y 100 bytes salida:" >> time.txt
(time ./tp1 -i entrada.txt -o salida.txt -I 1 -O 100)2>>time.txt
diff salida.txt resultado.txt

echo -e "\nResultado con 100 bytes entrada y 1 byte salida:" >> time.txt
(time ./tp1 -i entrada.txt -o salida.txt -I 100 -O 1)2>>time.txt
diff salida.txt resultado.txt

echo -e "\nResultado con 20 bytes entrada y 1000 bytes salida:" >> time.txt
(time ./tp1 -i entrada.txt -o salida.txt -I 20 -O 1000)2>>time.txt
diff salida.txt resultado.txt
```

```
50
51 echo -e "\nResultado con 1000 bytes entrada y 20 bytes salida:" >> time.txt
52 (time ./tp1 -i entrada.txt -o salida.txt -I 1000 -O 20)2>>time.txt
53 diff salida.txt resultado.txt
54
55 # Prueba con archivo vacio
56 touch vacio.txt
57 touch resultado_vacio.txt
58 ./tp1 -i vacio.txt -o salida.txt -I 1 -O 1
59 diff salida.txt resultado_vacio.txt
60 ./tp1 -i vacio.txt -o salida.txt -I 20 -O 20
61 diff salida.txt resultado_vacio.txt
62 ./tp1 -i vacio.txt -o salida.txt -I 100 -O 100
63 diff salida.txt resultado_vacio.txt
64 ./tp1 -i vacio.txt -o salida.txt -I 1000 -O 1000
65 diff salida.txt resultado_vacio.txt
66 ./tp1 -i vacio.txt -o salida.txt -I 20 -O 100
67 diff salida.txt resultado_vacio.txt
68 ./tp1 -i vacio.txt -o salida.txt -I 100 -O 20
69 diff salida.txt resultado_vacio.txt
70 ./tp1 -i vacio.txt -o salida.txt -I 1 -O 100
71 diff salida.txt resultado_vacio.txt
72 ./tp1 -i vacio.txt -o salida.txt -I 100 -O 1
73 diff salida.txt resultado_vacio.txt
74 ./tp1 -i vacio.txt -o salida.txt -I 20 -O 1000
75 diff salida.txt resultado_vacio.txt
76 ./tp1 -i vacio.txt -o salida.txt -I 1000 -O 20
77 diff salida.txt resultado_vacio.txt
78
79 # Pruebas con una sola letra mayúscula
80 echo M > res.txt
81 echo M | ./tp1 -o salida.txt -I 1 -O 1
82 diff salida.txt res.txt
83 echo M | ./tp1 -o salida.txt -I 20 -O 20
84 diff salida.txt res.txt
85 echo M | ./tp1 -o salida.txt -I 100 -O 100
86 diff salida.txt res.txt
87 echo M | ./tp1 -o salida.txt -I 1000 -O 1000
88 diff salida.txt res.txt
89 echo M | ./tp1 -o salida.txt -I 20 -O 100
90 diff salida.txt res.txt
91 echo M | ./tp1 -o salida.txt -I 100 -O 20
92 diff salida.txt res.txt
93 echo M | ./tp1 -o salida.txt -I 1 -O 100
94 diff salida.txt res.txt
95 echo M | ./tp1 -o salida.txt -I 100 -O 1
96 diff salida.txt res.txt
97 echo M | ./tp1 -o salida.txt -I 20 -O 1000
98 diff salida.txt res.txt
99 echo M | ./tp1 -o salida.txt -I 1000 -O 20
100 diff salida.txt res.txt
101
102 # Pruebas con una sola letra minúscula
```

```
103 echo m > res.txt
104 echo m | ./tp1 -o salida.txt -I 1 -O 1
105 diff salida.txt res.txt
106 echo m | ./tp1 -o salida.txt -I 20 -O 20
107 diff salida.txt res.txt
108 echo m | ./tp1 -o salida.txt -I 100 -O 100
109 diff salida.txt res.txt
110 echo m | ./tp1 -o salida.txt -I 1000 -O 1000
111 diff salida.txt res.txt
112 echo m | ./tp1 -o salida.txt -I 20 -O 100
113 diff salida.txt res.txt
114 echo m | ./tp1 -o salida.txt -I 100 -O 20
115 diff salida.txt res.txt
116 echo m | ./tp1 -o salida.txt -I 1 -O 100
117 diff salida.txt res.txt
118 echo m | ./tp1 -o salida.txt -I 100 -O 1
119 diff salida.txt res.txt
120 echo m | ./tp1 -o salida.txt -I 20 -O 1000
121 diff salida.txt res.txt
122 echo m | ./tp1 -o salida.txt -I 1000 -O 20
123 diff salida.txt res.txt
124
125 # Prueba con un número
126 echo 3 > res.txt
127 echo 3 | ./tp1 -o salida.txt -I 1 -O 1
128 diff salida.txt res.txt
129 echo 3 | ./tp1 -o salida.txt -I 20 -O 20
130 diff salida.txt res.txt
131 echo 3 | ./tp1 -o salida.txt -I 100 -O 100
132 diff salida.txt res.txt
133 echo 3 | ./tp1 -o salida.txt -I 1000 -O 1000
134 diff salida.txt res.txt
135 echo 3 | ./tp1 -o salida.txt -I 20 -O 100
136 diff salida.txt res.txt
137 echo 3 | ./tp1 -o salida.txt -I 100 -O 20
138 diff salida.txt res.txt
139 echo 3 | ./tp1 -o salida.txt -I 1 -O 100
140 diff salida.txt res.txt
141 echo 3 | ./tp1 -o salida.txt -I 100 -O 1
142 diff salida.txt res.txt
143 echo 3 | ./tp1 -o salida.txt -I 20 -O 1000
144 diff salida.txt res.txt
145 echo 3 | ./tp1 -o salida.txt -I 1000 -O 20
146 diff salida.txt res.txt
147
148 # Pruebas con un guion
149 echo - > res.txt
150 echo - | ./tp1 -o salida.txt -I 1 -O 1
151 diff salida.txt res.txt
152 echo - | ./tp1 -o salida.txt -I 20 -O 20
153 diff salida.txt res.txt
154 echo - | ./tp1 -o salida.txt -I 100 -O 100
155 diff salida.txt res.txt
```

```
156 echo - | ./tp1 -o salida.txt -I 1000 -O 1000
157 diff salida.txt res.txt
158 echo - | ./tp1 -o salida.txt -I 20 -O 100
159 diff salida.txt res.txt
160 echo - | ./tp1 -o salida.txt -I 100 -O 20
161 diff salida.txt res.txt
162 echo - | ./tp1 -o salida.txt -I 1 -O 100
163 diff salida.txt res.txt
164 echo - | ./tp1 -o salida.txt -I 100 -O 1
165 diff salida.txt res.txt
166 echo - | ./tp1 -o salida.txt -I 20 -O 1000
167 diff salida.txt res.txt
168 echo - | ./tp1 -o salida.txt -I 1000 -O 20
169 diff salida.txt res.txt
170
171 # Pruebas con un guion bajo
172 echo _ > res.txt
173 echo _ | ./tp1 -o salida.txt -I 1 -O 1
174 diff salida.txt res.txt
175 echo _ | ./tp1 -o salida.txt -I 20 -O 20
176 diff salida.txt res.txt
177 echo _ | ./tp1 -o salida.txt -I 100 -O 100
178 diff salida.txt res.txt
179 echo _ | ./tp1 -o salida.txt -I 1000 -O 1000
180 diff salida.txt res.txt
181 echo _ | ./tp1 -o salida.txt -I 20 -O 100
182 diff salida.txt res.txt
183 echo _ | ./tp1 -o salida.txt -I 100 -O 20
184 diff salida.txt res.txt
185 echo _ | ./tp1 -o salida.txt -I 1 -O 100
186 diff salida.txt res.txt
187 echo _ | ./tp1 -o salida.txt -I 100 -O 1
188 diff salida.txt res.txt
189 echo _ | ./tp1 -o salida.txt -I 20 -O 1000
190 diff salida.txt res.txt
191 echo _ | ./tp1 -o salida.txt -I 1000 -O 20
192 diff salida.txt res.txt
193
194 # Pruebas con un simbolo
195 echo @ | ./tp1 -o salida.txt -I 1 -O 1
196 diff salida.txt vacio.txt
197 echo @ | ./tp1 -o salida.txt -I 20 -O 20
198 diff salida.txt vacio.txt
199 echo @ | ./tp1 -o salida.txt -I 100 -O 100
200 diff salida.txt vacio.txt
201 echo @ | ./tp1 -o salida.txt -I 1000 -O 1000
202 diff salida.txt vacio.txt
203 echo @ | ./tp1 -o salida.txt -I 20 -O 100
204 diff salida.txt vacio.txt
205 echo @ | ./tp1 -o salida.txt -I 100 -O 20
206 diff salida.txt vacio.txt
207 echo @ | ./tp1 -o salida.txt -I 1 -O 100
208 diff salida.txt vacio.txt
```

```
209 echo @ | ./tp1 -o salida.txt -I 100 -O 1
210 diff salida.txt vacio.txt
211 echo @ | ./tp1 -o salida.txt -I 20 -O 1000
212 diff salida.txt vacio.txt
213 echo @ | ./tp1 -o salida.txt -I 1000 -O 20
214 diff salida.txt vacio.txt
215
216 # Prueba con espacios
217 echo "                    " > ent.txt
218 ./tp1 -i ent.txt -o salida.txt -I 1 -O 1
219 diff salida.txt vacio.txt
220 ./tp1 -i ent.txt -o salida.txt -I 20 -O 20
221 diff salida.txt vacio.txt
222 ./tp1 -i ent.txt -o salida.txt -I 100 -O 100
223 diff salida.txt vacio.txt
224 ./tp1 -i ent.txt -o salida.txt -I 1000 -O 1000
225 diff salida.txt vacio.txt
226 ./tp1 -i ent.txt -o salida.txt -I 20 -O 100
227 diff salida.txt vacio.txt
228 ./tp1 -i ent.txt -o salida.txt -I 100 -O 20
229 diff salida.txt vacio.txt
230 ./tp1 -i ent.txt -o salida.txt -I 1 -O 100
231 diff salida.txt vacio.txt
232 ./tp1 -i ent.txt -o salida.txt -I 100 -O 1
233 diff salida.txt vacio.txt
234 ./tp1 -i ent.txt -o salida.txt -I 20 -O 1000
235 diff salida.txt vacio.txt
236 ./tp1 -i ent.txt -o salida.txt -I 1000 -O 20
237 diff salida.txt vacio.txt
238
239 # Pruebas con simbolos
240 echo "@#$%^*()!{}[],./?<>;:*+\|=+" > ent.txt
241 ./tp1 -i ent.txt -o salida.txt -I 1 -O 1
242 diff salida.txt vacio.txt
243 ./tp1 -i ent.txt -o salida.txt -I 20 -O 20
244 diff salida.txt vacio.txt
245 ./tp1 -i ent.txt -o salida.txt -I 100 -O 100
246 diff salida.txt vacio.txt
247 ./tp1 -i ent.txt -o salida.txt -I 1000 -O 1000
248 diff salida.txt vacio.txt
249 ./tp1 -i ent.txt -o salida.txt -I 20 -O 100
250 diff salida.txt vacio.txt
251 ./tp1 -i ent.txt -o salida.txt -I 100 -O 20
252 diff salida.txt vacio.txt
253 ./tp1 -i ent.txt -o salida.txt -I 1 -O 100
254 diff salida.txt vacio.txt
255 ./tp1 -i ent.txt -o salida.txt -I 100 -O 1
256 diff salida.txt vacio.txt
257 ./tp1 -i ent.txt -o salida.txt -I 20 -O 1000
258 diff salida.txt vacio.txt
259 ./tp1 -i ent.txt -o salida.txt -I 1000 -O 20
260 diff salida.txt vacio.txt
261
```

```
262
263 #Prueba con un archivo con 30 lineas de 5000 caracteres cada una,
264 # donde cada una es palindromo en su totalidad
265 echo -e "\nPRUEBAS CON ARCHIVO CON LINEAS DE 5000 CARACTERES CADA UNA (
      TODAS SON PALINDROMO)\n" >> time.txt
266
267 echo -e "\nResultado con 1 byte entrada y 1 byte salida:" >> time.txt
268 (time ./tp1 -i archivo_largo.txt -o salida.txt -I 1 -O 1)2>>time.txt
269 diff salida.txt archivo_largo.txt
270
271 echo -e "\nResultado con 20 bytes entrada y 20 bytes salida:" >> time.txt
272 (time ./tp1 -i archivo_largo.txt -o salida.txt -I 20 -O 20)2>>time.txt
273 diff salida.txt archivo_largo.txt
274
275 echo -e "\nResultado con 100 bytes entrada y 100 bytes salida:" >> time.txt
276 (time ./tp1 -i archivo_largo.txt -o salida.txt -I 100 -O 100)2>>time.txt
277 diff salida.txt archivo_largo.txt
278
279 echo -e "\nResultado con 1000 bytes entrada y 1000 bytes salida:" >> time.
      txt
280 (time ./tp1 -i archivo_largo.txt -o salida.txt -I 1000 -O 1000)2>>time.txt
281 diff salida.txt archivo_largo.txt
282
283 echo -e "\nResultado con 20 bytes entrada y 100 bytes salida:" >> time.txt
284 (time ./tp1 -i archivo_largo.txt -o salida.txt -I 20 -O 100)2>>time.txt
285 diff salida.txt archivo_largo.txt
286
287 echo -e "\nResultado con 100 bytes entrada y 20 bytes salida:" >> time.txt
288 (time ./tp1 -i archivo_largo.txt -o salida.txt -I 100 -O 20)2>>time.txt
289 diff salida.txt archivo_largo.txt
290
291 echo -e "\nResultado con 1 byte entrada y 20 bytes salida:" >> time.txt
292 (time ./tp1 -i archivo_largo.txt -o salida.txt -I 1 -O 100)2>>time.txt
293 diff salida.txt archivo_largo.txt
294
295 echo -e "\nResultado con 100 bytes entrada y 1 byte salida:" >> time.txt
296 (time ./tp1 -i archivo_largo.txt -o salida.txt -I 100 -O 1)2>>time.txt
297 diff salida.txt archivo_largo.txt
298
299 echo -e "\nResultado con 20 bytes entrada y 1000 bytes salida:" >> time.txt
300 (time ./tp1 -i archivo_largo.txt -o salida.txt -I 20 -O 1000)2>>time.txt
301 diff salida.txt archivo_largo.txt
302
303 echo -e "\nResultado con 1000 bytes entrada y 20 bytes salida:" >> time.txt
304 (time ./tp1 -i archivo_largo.txt -o salida.txt -I 1000 -O 20)2>>time.txt
305 diff salida.txt archivo_largo.txt
306
307 # Prueba error: no se ingresa archivo de entrada
308 echo "Debe indicar un archivo de entrada luego de -i" > res.txt
309 ./tp1 -i 2> error.txt
310 diff error.txt res.txt
311 ./tp1 -I 10 -i 2> error.txt
312 diff error.txt res.txt
```

```
313 ./tp1 -I 10 -O 10 -i 2> error.txt
314 diff error.txt res.txt
315 ./tp1 -o salida.txt -i 2> error.txt
316 diff error.txt res.txt
317
318 # Prueba error: no se ingresa archivo de salida
319 echo "Debe indicar un archivo de salida luego de -o" > res.txt
320 ./tp1 -o 2> error.txt
321 diff error.txt res.txt
322 ./tp1 -i entrada.txt -o 2> error.txt
323 diff error.txt res.txt
324 ./tp1 -I 10 -o 2> error.txt
325 diff error.txt res.txt
326 ./tp1 -I 10 -O 10 -o 2> error.txt
327 diff error.txt res.txt
328
329 # Prueba error: no se ingresa tamaño del buffer de entrada
330 echo "Debe indicar un numero luego de -I" > res.txt
331 ./tp1 -I 2> error.txt
332 diff error.txt res.txt
333 ./tp1 -i entrada.txt -I 2> error.txt
334 diff error.txt res.txt
335 ./tp1 -O 10 -I 2> error.txt
336 diff error.txt res.txt
337 ./tp1 -i entrada.txt -O 10 -I 2> error.txt
338 diff error.txt res.txt
339
340 # Prueba error: no se ingresa tamaño del buffer de salida
341 echo "Debe indicar un numero luego de -O" > res.txt
342 ./tp1 -O 2> error.txt
343 diff error.txt res.txt
344 ./tp1 -i entrada.txt -O 2> error.txt
345 diff error.txt res.txt
346 ./tp1 -I 10 -O 2> error.txt
347 diff error.txt res.txt
348 ./tp1 -i entrada.txt -I 10 -O 2> error.txt
349 diff error.txt res.txt
350
351
352 # Prueba error: no se puede abrir el archivo de entrada
353 echo "El archivo de entrada no pudo abrirse" > res.txt
354 ./tp1 -i inexistente.txt 2> error.txt
355 diff error.txt res.txt
356 ./tp1 -o salida.txt -i inexistente.txt 2> error.txt
357 diff error.txt res.txt
358 ./tp1 -I 10 -i inexistente.txt 2> error.txt
359 diff error.txt res.txt
360 ./tp1 -i inexistente.txt -I 10 2> error.txt
361 diff error.txt res.txt
362
363 # Prueba error: el tamaño del buffer de entrada no es un numero
364 echo "El parametro de -I debe ser un numero" > res.txt
365 ./tp1 -I abc 2> error.txt
```

```
366 diff error.txt res.txt
367 ./tp1 -i entrada.txt -I numero 2> error.txt
368 diff error.txt res.txt
369 ./tp1 -O 10 -I nueve 2> error.txt
370 diff error.txt res.txt
371 ./tp1 -i entrada.txt -O 10 -I abc123 2> error.txt
372 diff error.txt res.txt
373
374 # Prueba error: el tamaño del buffer de salida no es un numero
375 echo "El parametro de -O debe ser un numero" > res.txt
376 ./tp1 -O abc 2> error.txt
377 diff error.txt res.txt
378 ./tp1 -i entrada.txt -O numero 2> error.txt
379 diff error.txt res.txt
380 ./tp1 -I 10 -O nueve 2> error.txt
381 diff error.txt res.txt
382 ./tp1 -i entrada.txt -I 10 -O abc123 2> error.txt
383 diff error.txt res.txt
384
385 #Pruebas con stdin (sin poner '-i' o poniendo '-i -')
386 ./tp1 -o salida.txt -I 1 -O 1  < entrada.txt
387 diff salida.txt resultado.txt
388 ./tp1 -o salida.txt -I 20 -O 20  < entrada.txt
389 diff salida.txt resultado.txt
390 ./tp1 -o salida.txt -I 100 -O 100  < entrada.txt
391 diff salida.txt resultado.txt
392 ./tp1 -o salida.txt -I 1000 -O 1000  < entrada.txt
393 diff salida.txt resultado.txt
394 ./tp1 -o salida.txt -I 20 -O 100  < entrada.txt
395 diff salida.txt resultado.txt
396 ./tp1 -o salida.txt -I 100 -O 20  < entrada.txt
397 diff salida.txt resultado.txt
398 ./tp1 -i - -o salida.txt -I 1 -O 100  < entrada.txt
399 diff salida.txt resultado.txt
400 ./tp1 -i - -o salida.txt -I 100 -O 1  < entrada.txt
401 diff salida.txt resultado.txt
402 ./tp1 -i - -o salida.txt -I 20 -O 1000  < entrada.txt
403 diff salida.txt resultado.txt
404 ./tp1 -i - -o salida.txt -I 1000 -O 20  < entrada.txt
405 diff salida.txt resultado.txt
406
407
408 #Prueba con stdout (sin poner '-o' o poniendo '-o -')
409 ./tp1 -i entrada.txt -I 1 -O 1 > salida.txt
410 diff salida.txt resultado.txt
411 ./tp1 -i entrada.txt -I 20 -O 20 > salida.txt
412 diff salida.txt resultado.txt
413 ./tp1 -i entrada.txt -I 100 -O 100 > salida.txt
414 diff salida.txt resultado.txt
415 ./tp1 -i entrada.txt -I 1000 -O 1000 > salida.txt
416 diff salida.txt resultado.txt
417 ./tp1 -i entrada.txt -I 20 -O 100 > salida.txt
418 diff salida.txt resultado.txt
```

```
./tp1 -i entrada.txt -I 100 -O 20 > salida.txt
diff salida.txt resultado.txt
./tp1 -i entrada.txt -I 1 -O 100 > salida.txt
diff salida.txt resultado.txt
./tp1 -i entrada.txt -I 100 -O 1 > salida.txt
diff salida.txt resultado.txt
./tp1 -i entrada.txt -I 20 -O 1000 > salida.txt
diff salida.txt resultado.txt
./tp1 -i entrada.txt -I 1000 -O 20 > 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
```

3.4. Archivo archivo_largo.txt

La función de este archivo es trabajar con varios palíndromos largos para así poder apreciar la diferencia en los tiempos de ejecución. A continuación se muestra la primer línea de este archivo. Las lineas faltantes son una copia de ésta.

ElavidolocopaocopleropodiaHablaorecitayarmamasacadadiaSeamorysiesamor
brevesaletodotanensubonitaparteparalamiradadelosotrosyosadulaseramada
solamasleneEvaySaradanelbondadosodonaseresroponesyodiososUlisesilusos
eranagriosAnaSaritaLaraconavidalocasoledadEvaElsalaotraPetrasonamigas
LafacilocubanaSaritasevaapartaaraleaconotroBasilioSeissonamasamareson
inadamamalarotasosamonaTigresyososresidenopacosunosalacazarotadelamad
onayotrosbalanalegresLasamadasnacenenlasaridasuralesrocasUnaeslarusaM
irapordepararyanadirapocolaligaoyalacalaDeseonaZolaterajeRosaesolalam
inalamateMoloanaziuranioRehusorimelenodiosoojoLamejorrazaparaamarosal
ejatobasosamasinevasivasAmorbabososiesnoserahoyamorAledenicodasanapar
aplatonicotemaAsurarteplacidonadamodeladoLamoritaLaraseabocadadesimas
eseaveacalopalioylaseparosanomisyatiVeedadIvanasusamaritanaelevalosse
nosAnaropasacaManolocedeLedicededucalesamoresadoropodernenaserimanago
reroTorotasadoyahoracorneadoeselerigidocomoelajadoNoasonodonusoreyEra
honortapaduraosadaOdotedarodaleverYasepisanRutconnatadaredopadaDeloso
loSalnenemetraesonAmaralysolracionadoPetranoconocesolyoirfoliaconIgor
opinanegadoranegadamareartesoloyabusoEsedalocacocainaMortodasoledadEs
everynoseresodioOjocaefataloroModadopaOilimeNococinaTasotiradobusnore
idrusodaraCansenAliVagonorailalsonUnabalacalabasonAgroserometiosolaMo
jonEsamasleoyacosodeimitadasaperturasonomenoseroticasIramalevolaTamar
alamalaalososadosatacoperonoElsamaridososogujocoponysosotitepasonivid
aNosasolalasodasAseramalacosoRasolyzorraosorbasepaganargarboselamoTom
amadreimanagasOiledadalenotraceproyamorolodorarropaNacaradotioconsies
onoesaellasyavalenorocaladasEpopeyarojabarataOirleAnaesoleajedelocaso
catalanasaneherederaHoynomajaMirelavadotadatodavaradalaligaabajoManol
oidemyatalamotiraronelpavoracaloradoRomanavinomilanesaoiResenabemolep
idemiasamosatenaSolacolarapielesorasuraraperrasApaticaseesEsoidoNorep
aresOiDomoconsumonimoderadonisacroLodominaradesderaboacaboSonaramosal
aneveramosaicocolumnaSiUgaldedecorominimosAparemujereslaElsamaslealla
SaramiralamarAdemasellasenelorbelecomeranellevellanodelociosovaronete
reoroedoryamargadopaladinAsorgenOsoirasosovelanaresEressoleladem anoso
sivaadelatarosodiraotraharinamoleranimaCaminarasegurodesusabaticonatu
ralNenasosanunisonasElloslesadornanunoslacaraperodanotrosalugaresocas
ipululanenasediosinunapicemasamigoCarameloyagrioledieronysupomalAsumi
rehoynosolodiahoratodoesedeterioroMaladiosupersonaRimerotasaparabolas
enanasysinueraesaanulaloriconoconocesecorefranagalerasaremararemaraga
lerasatoroyyermosomreyyorotasarelagarameraramerasarelaganarferoceseco
noconocirolalunaaseareunisysananesalobarapasatoremiRanosrepusoidalaMo
roiretedeseodotarohaidolosonyoherimusAlamopusynoreideloirgayolemaraCo
gimasamecipanunisoidesanenalulupisacoseragulasortonadoreparacalsonuna
nrodaselsollEsanosinunasosaneNlarutanocitabasusedorugesaranimaCaminar
elomanirahartoaridosorataledaavisosonamedalelosserEseranalevososarios
OnegrosAnidalapodagramayrodeoroeretenoravosoicoledonalLevelLenaremoce
lebrolenesallesamedAramalarimaraSallaelsamaslEalserejumerapAsominimor
ocededlagUiSanmulocociasomarevenalasomaranoSobacaobaredsedaranimodoLo
rcasinodaredominomusnocomoDiOseraperoNodiosEseesacitapAsarrepararusar
oseleiparalocaloSanetasomasaimedipelomebaneseRioasenalimonivanamoRoda
rolacarovaplenoraritomalataymediolonaMojabaagilaladaravadotadatodaval
eriMajamonyoHarederehenasanalatacosacoledejaeloseanAelriOatarabajoray

epopEsadalacoronelavaysalleaseonoseisnocoitodaracaNaporrarodoloromayo
rpecartoneladadeliOsaganamierdamamoTomalesobragranagapesabrosoarrozyl
osaRosocalamaresAsadosalalosasoNadivinosapetitososynopocojugososodira
maslEonorepocatasodasosolaalamalaramaTalovelamarIsacitoresonemonosaru
trepasadatimiedosocayoelsamasEnojoMalosoitemoresorgAnosabalacalabanUn
oslaliaronogaVilAnesnaCaradosurdieronsubodaritosaTanicocoNemiliOapoda
doMorolatafeacojOoidoseresonyrevesEdadelosadotroManiacocacoladesEosub
ayolosetraeramadagenarodagenaniporogInocailofrioyloseconoconartePodan
oicarlosylaramAnoseartemenenlaSolosoleDadapoderadatannoctuRnasipesaYr
eveladoradetodOadasoarudapatronoharEyerosunodonosaoNodajaleomocodigir
eleseodaenrocarohayodasatoroToreroganamiresanenredoporodaseromaselacu
dedecideLedecolonaMacasaporanAsonessolaveleanatiramasusanavIdadeeVita
ysimonasorapesalyoilapolacaevaesesamisedadacobaesaraLatiromaLodaledom
adanodicalpetrarusAametocinotalparapanasadocinedelAromayoharesonseiso
sobabromAsavisavenisamasosabotajelasoramaarapazarrojemaLojoosoidonele
mirosuheRoinaruizanaoloMetamalanimalaloseasoRejaretaloZanoeseDalacala
yoagilalocoparidanayrarapedropariMasuralseanUsacorselarusadirasalnene
cansadamasaLsergelanalabsortoyanodamaledatorazacalasonusocaponedisers
osoysergiTanomasosatoralamamadaninoseramasamanossieSoilisaBortonocael
araatrapaavesatiraSanabucolicafaLsagimanosartePartoalaslEavEdadelosac
oladivanocaraLatiraSanAsoirganaresosulisesilUsosoidoysenoporseresanod
osodadnoblenadaraSyavEenelsamalosadamaresaludasoysortosoledadarimalar
apetrapatinobusnenatodotelaseverbromaseisyromaeSaidadacasamamrayatice
roAlbahaidoporelPocoapocolodiVale

3.5. Archivo time.txt

PRUEBAS CON ARCHIVO DE PRUEBAS ENTRADA.TXT Y RESULTADO.TXT

```
Resultado con 1 byte entrada y 1 byte salida:
        0.10 real         0.03 user         0.10 sys

Resultado con 20 bytes entrada y 20 bytes salida:
        0.06 real         0.00 user         0.07 sys

Resultado con 100 bytes entrada y 100 bytes salida:
        0.04 real         0.03 user         0.04 sys

Resultado con 1000 bytes entrada y 1000 bytes salida:
        0.05 real         0.03 user         0.03 sys

Resultado con 20 bytes entrada y 100 bytes salida:
        0.06 real         0.00 user         0.08 sys

Resultado con 100 bytes entrada y 20 bytes salida:
        0.05 real         0.02 user         0.05 sys

Resultado con 1 byte entrada y 100 bytes salida:
        0.08 real         0.03 user         0.06 sys
```

```
Resultado con 100 bytes entrada y 1 byte salida:
        0.08 real         0.02 user         0.09 sys

Resultado con 20 bytes entrada y 1000 bytes salida:
        0.04 real         0.02 user         0.05 sys

Resultado con 1000 bytes entrada y 20 bytes salida:
        0.06 real         0.02 user         0.05 sys

PRUEBAS CON ARCHIVO CON LINEAS DE 5000 CARACTERES CADA UNA (TODAS SON PALINDROMO)


Resultado con 1 byte entrada y 1 byte salida:
       19.54 real         2.39 user        17.13 sys

Resultado con 20 bytes entrada y 20 bytes salida:
        3.60 real         2.28 user         1.34 sys

Resultado con 100 bytes entrada y 100 bytes salida:
        2.89 real         2.23 user         0.67 sys

Resultado con 1000 bytes entrada y 1000 bytes salida:
        2.76 real         2.24 user         0.53 sys

Resultado con 20 bytes entrada y 100 bytes salida:
        3.08 real         2.32 user         0.78 sys

Resultado con 100 bytes entrada y 20 bytes salida:
        3.41 real         2.20 user         1.22 sys

Resultado con 1 byte entrada y 20 bytes salida:
        7.19 real         2.59 user         4.60 sys

Resultado con 100 bytes entrada y 1 byte salida:
       15.26 real         2.29 user        12.98 sys

Resultado con 20 bytes entrada y 1000 bytes salida:
        2.96 real         2.25 user         0.73 sys

Resultado con 1000 bytes entrada y 20 bytes salida:
        3.36 real         2.23 user         1.12 sys
```

4. Main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 extern int palindrome (int ifd, size_t ibytes, int ofd, size_t obytes);
6
7 int main(int argc, char* argv[]){
8     FILE* entrada = stdin;
9     FILE* salida = stdout;
```

```
10      int tam_buffer_entrada = 1;
11      int tam_buffer_salida = 1;
12      char* parametro;
13
14      int i;
15      for (i = 1; i < argc; i += 2){
16          if (strcmp(argv[i],"-i") == 0){
17              if (i + 1 >= argc){
18                  fputs("Debe indicar un archivo de entrada luego de -i\n",
                        stderr);
19                  return 3;
20              }
21              parametro = argv[i + 1];
22              if (strcmp(parametro,"-") != 0){
23                  entrada = fopen(argv[i + 1], "r");
24                  if (!entrada){
25                      fputs("El archivo de entrada no pudo abrirse\n", stderr
                            );
26                      return 4;
27                  }
28              }
29          }
30          else if (strcmp(argv[i],"-o") == 0){
31              if (i + 1 >= argc){
32                  fputs("Debe indicar un archivo de salida luego de -o\n",
                        stderr);
33                  return 3;
34              }
35              parametro = argv[i + 1];
36              if (strcmp(parametro,"-") != 0){
37                  salida = fopen(argv[i + 1], "w");
38                  if (!salida){
39                      fputs("El archivo de salida no pudo abrirse\n", stderr)
                            ;
40                      return 4;
41                  }
42              }
43          }
44          else if (strcmp(argv[i],"-I") == 0){
45              if (i + 1 >= argc){
46                  fputs("Debe indicar un numero luego de -I\n", stderr);
47                  return 3;
48              }
49              parametro = argv[i + 1];
50              if (strcmp(parametro,"-") != 0){
51                  tam_buffer_entrada = atoi(parametro);
52                  if (tam_buffer_entrada == 0){
53                      fputs("El parametro de -I debe ser un numero\n", stderr
                            );
54                      return 4;
55                  }
56              }
57          }
```

```
58          else if (strcmp(argv[i],"-O") == 0){
59              if (i + 1 >= argc){
60                  fputs("Debe indicar un numero luego de -O\n", stderr);
61                  return 3;
62              }
63              parametro = argv[i + 1];
64              if (strcmp(parametro,"-") != 0){
65                  tam_buffer_salida = atoi(parametro);
66                  if (tam_buffer_salida == 0){
67                      fputs("El parametro de -O debe ser un numero\n", stderr
                            );
68                      return 4;
69                  }
70              }
71          }
72          else if (strcmp(argv[i],"-V") == 0){
73              fprintf(stdout, "TP1 version 1.0001\n");
74              return 0;
75          }
76          else if (strcmp(argv[i],"-h") == 0){
77              fprintf(stdout, "Usage:\n\ntp1 -h\ntp1 -V\ntp1 [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-I, --ibuf-bytes Byte-count of the input buffer.\n-O, --
                    obuf-bytes Byte-count of the output buffer.\n\nExample:\ntp0
                    -i ~/input -o ~/output\n");
78              return 0;
79          }
80      }
81
82      int resultado = palindrome(fileno(entrada), tam_buffer_entrada, fileno(
            salida), tam_buffer_salida);
83
84      if (resultado != 0){
85          fputs("Ocurrio un error\n", stderr);
86      }
87
88      fclose(entrada);
89      fclose(salida);
90
91      return resultado;
92  }
```

5. Myrealloc.S

```
1  #include <mips/regdef.h>
2  #include <sys/syscall.h>
3
4  .text
5  .abicalls
6  .align 2
7
```

```
 8 .globl myrealloc
 9 .ent myrealloc
10
11 myrealloc:
12     .frame $fp, 40, ra
13     .set noreorder
14     .cpload t9
15     .set reorder
16
17     subu sp, sp, 40
18     sw s0, 24(sp)
19     sw $fp, 28(sp)
20     .cprestore 32
21     sw ra, 36(sp)
22     move $fp, sp
23
24     sw a0, 40($fp)                  # guardamos el puntero
25     sw a1, 44($fp)                  # guardamos el tamanio del bloque
26     sw a2, 48($fp)                  # guardamos el tamanio a agregar
27
28     addu a0, a1, a2                 #a0 = nuevo tamanio bloque
29     la t9, mymalloc
30     jalr t9                         #Llama a malloc
31     beq v0, zero, return            #Si devuelve cero ocurrio un error
           y devuelve cero
32
33     sw v0, 16($fp)                  #Guardamos el puntero nuevo
34     addu s0, zero, zero             #s0 = indice actual
35
36 loop:
37     lw t0, 44($fp)                  #t0 = tamanio original
38     bgeu s0, t0, terminar           #salta si ya copio todo
39
40     lw t0, 40 ($fp)                 #t0 = puntero viejo
41     addu t0, t0, s0                 #t0 = puntero viejo + indice
42
43     lw t1, 16($fp)                  #t1 = puntero nuevo
44     addu t1, t1, s0                 #t1 = puntero nuevo + indice
45
46     lbu t2, 0(t0)                   #Cargo el byte a copiar en t2
47     sb t2, 0(t1)                    #Guardo el byte
48
49     addiu s0, s0, 1                 #incremento el indice
50     b loop                          #Vuelve al loop
51
52 terminar:
53     lw a0, 40($fp)                  #a0 = puntero viejo
54     la t9, myfree
55     jalr t9                         #Libera el puntero viejo
56     lw v0, 16($fp)                  #v0 = puntero nuevo
57
58 return:
59     lw s0, 24($fp)
```

```
60      lw ra, 36($fp)
61      lw gp, 32($fp)
62      lw $fp, 28($fp)
63      addiu sp, sp, 40
64      jr ra
65
66  .end myrealloc
67  .size myrealloc,.-myrealloc
```

6. Palindrome.S

```
 1  #include <mips/regdef.h>
 2  #include <sys/syscall.h>
 3  #define ASCII_A_MAY 65
 4  #define ASCII_Z_MAY 90
 5  #define ASCII_A_MIN 97
 6  #define ASCII_Z_MIN 122
 7  #define ASCII_CERO 48
 8  #define ASCII_NUEVE 57
 9  #define ASCII_GUION 45
10  #define ASCII_GUIONBAJO 95
11  #define ASCII_NEWLINE 10
12
13  .text
14  .abicalls
15  .align 2
16
17  .globl palindrome
18  .ent palindrome
19
20  palindrome:
21      .frame $fp, 48, ra
22      .set noreorder
23      .cpload t9
24      .set reorder
25
26      subu sp, sp, 48
27      sw $fp, 32(sp)
28      .cprestore 36
29      sw ra, 40(sp)
30      move $fp, sp
31
32      sw a0, 48($fp)                  # guardamos el archivo de entrada
            en el stackframe
33      sw a1, 52($fp)                  # guardamos el tamanio buffer
            entrada
34      sw a2, 56($fp)                  # guardamos el archivo de salida en
             el stackframe
35      sw a3, 60($fp)                  # guardamos el tamanio buffer
            salida
36
37      move a0, a1                     #a0 = tamanio del buffer entrada
38
```

```
39      la t9, crear_buffer
40      jalr t9                             #Crea el buffer de entrada
41      beq v0, zero, error_primer_buffer   #Si devuelve 0 ocurrio un error
42      sw v0, 24($fp)                      #Guardamos el buffer de entrada en
           el stackframe
43      la t0, tam_buffer_entrada
44      lw t1, 52($fp)
45      sw t1, 0(t0)                        #Actualizo el tamanio del buffer
           entrada
46
47      lw a0, 60($fp)                      #a0 = tamanio del buffer de salida
48      la t9, crear_buffer
49      jalr t9                             #Crea el buffer de salida
50      beq v0, zero, error_segundo_buffer  #Si devuelve 0 ocurrio un error
51      sw v0, 28($fp)                      #Guardamos el buffer de salida en
           el stackframe
52      la t0, tam_buffer_salida
53      lw t1, 60($fp)
54      sw t1, 0(t0)                        #Actualizo el tamanio del buffer
           salida
55
56  loop_palindrome:
57      lw a0, 48($fp)                      #Cargamos en a0 el archivo de
           entrada
58      addu a1, $fp, 16                    #Cargamos en a1 el puntero a len
59      lw a2, 24($fp)                      #Cargamos en a2 el buffer de
           entrada
60
61      la t9, leer_palabra
62      jalr t9                             #Lee la proxima palabra
63
64      sw v0, 20($fp)                      # Guardamos el puntero a la palabra
65      beq v0, zero, error_palindrome      # No se pudo leer la palabra
66
67      move a0, v0                         #a0 = puntero a palabra
68      lw a1, 16 ($fp)                     #a1 = len(palabra)
69      la t9, es_capicua
70      jalr t9                             #Llama a es_capicua
71
72      beq v0, zero, continuar_loop_palindrome        #Si no es capicua
           continua el loop
73
74      lw a0, 56($fp)                      #a0 = archivo de salida
75      lw a1, 20($fp)                      #a1 = puntero a la palabra
76      lw a2, 28($fp)                      #a2 = buffer de salida
77      la t9, putch
78      jalr t9                             #Escribe la palabra en el archivo
79      beq v0, zero, error_escritura       #Si devuelve 0 ocurrio un error
80
81  continuar_loop_palindrome:
82
83      lw a0, 20($fp)                      #a0 = puntero a palabra
84      la t9, myfree
```

```
85      jalr t9                         #Llama a free(palabra)
86
87      la t0, eof_leido
88      lw t1, 0(t0)                    #t1 = eof leido?
89      beq t1, zero, loop_palindrome   #si no fue leido sigue el loop
90      la t0, eof_escrito
91      addiu t1, zero, 1
92      sw t1, 0(t0)                    #actualizo eof_escrito = 1
93      lw a0, 56($fp)                  #a0 = archivo de salida
94      lw a2, 28($fp)                  #a2 = buffer de salida
95      la t9, putch
96      jalr t9                         #Escribe todo lo que queda en el
           buffer
97      beq v0, zero, error_escritura   #Si devuelve 0 ocurrio un error
98
99  terminar_palindrome:
100     move v0, zero
101 free_segundo_buffer:
102     lw a0, 28($fp)                  #Cargamos el buffer de salida
103     la t9,myfree
104     jalr t9                         #Liberamos el segundo buffer
105 free_primer_buffer:
106     lw a0, 24($fp)                  #Cargamos el buffer de entrada
107     la t9,myfree
108     jalr t9                         #Liberamos el primer buffer
109
110 return_palindrome:
111     lw ra, 40($fp)
112     lw gp, 36($fp)
113     lw $fp, 32($fp)
114     addiu sp, sp, 48
115     jr ra
116
117 error_segundo_buffer:
118     addiu v0, zero, 1               #Devuelve codigo de error 1
119     b free_primer_buffer            #Libera el primer buffer
120 error_primer_buffer:
121     addiu v0, zero, 1               #Devuelve codigo de error 1
122     b return_palindrome             #No libera nada
123
124 error_escritura:
125     lw a0, 20($fp)                  #a0 = puntero a la palabra
126     la t9, myfree
127     jalr t9                         #Libera la palabra
128 error_palindrome:
129     addiu v0, zero, 2               #Devuelve codigo de error 2
130     b free_segundo_buffer           #Libera ambos buffer
131
132 .end palindrome
133 .size palindrome,.-palindrome
134
135 #
```

```
136
137 .ent es_capicua
138
139 es_capicua:
140     .frame $fp, 48, ra
141     .set noreorder
142     .cpload t9
143     .set reorder
144
145     subu sp, sp, 48
146     sw s0, 24(sp)
147     sw s1, 28(sp)
148     sw $fp, 32(sp)
149     .cprestore 36
150     sw ra, 40(sp)
151     move $fp, sp
152
153     sw a0, 48($fp)                  # Guardamos el puntero a la palabra (
            primer argumento) en el stackframe
154     sw a1, 52($fp)                  # Guardamos la longitud (segundo
            argumento) en el stackframe
155
156     beq a1, zero, return_false_capicua      #Si len es 0 devuelve false
157
158     addu s0, zero, zero             #Inicializamos la variable inicio
159     subu s1, a1, 1                  #Inicializamos la variable final
160
161 loop_capicua:
162     lw t0, 48($fp)          #Recuperamos el puntero a la palabra
163     addu t0, t0, s0         #t0 = palabra + inicio
164     lbu a0, 0(t0)           # Leemos el  caracter palabra[inicio]
165     la t9, my_tolower
166     jalr t9                 #Llamamos a tolower() con el caracter
167     sw v0, 16($fp)          # Guardamos el primer caracter en minuscula
168     lw a0, 48($fp)          #Recuperamos el puntero a la palabra
169     addu t0, a0, s1         #t0 = palabra + final
170     lbu a0, 0(t0)           # Leemos el  caracter palabra[final]
171     la t9, my_tolower
172     jalr t9                 #Llamamos a tolower() con el caracter
173     sw v0, 20($fp)          # Guardamos el segundo caracter en minuscula
174     move t0, v0             #t0 = segundo caracter
175     lw t1, 16($fp)          #t1 = primer caracter
176     bne t0, t1, return_false_capicua   #Si son distintos devuelve false
177
178     addiu s0, s0, 1         #Suma uno (un byte) a inicio
179     subu s1, s1, 1          #Resta uno (un byte) a final
180     blt s0, s1, loop_capicua        #Si inicio < final vuelve al loop
181
182 return_true_capicua:
183     addiu v0, zero, 1
184     b return_capicua
185
```

```
186 return_false_capicua:
187     addu v0, zero, zero
188
189 return_capicua:
190     lw ra, 40($fp)
191     lw gp, 36($fp)
192     lw s1, 28($fp)
193     lw s0, 24($fp)
194     lw $fp, 32($fp)
195     addiu sp, sp, 48
196     jr ra
197
198 .end es_capicua
199 .size es_capicua,.-es_capicua
200
201
202
203 #
    ------------------------------------------------------------------------------
204
205
206 .ent my_tolower
207
208 my_tolower:
209     .frame $fp, 24, ra
210     .set noreorder
211     .cpload t9
212     .set reorder
213
214     subu sp, sp, 24
215     sw $fp, 16(sp)
216     .cprestore 20
217     move $fp, sp
218
219     sw a0, 24($fp)              #Guardamos el primer argumento (caracter)
        en el stackframe
220
221     blt a0, ASCII_A_MAY, return_tolower     # Salta si caracter es menor a
        A
222     bgt a0, ASCII_Z_MAY, return_tolower     # Salta si caracter es mayor a
        Z
223     addiu a0, a0, 32                    # 32 es la diferencia entre
        minusculas y mayusculas en ascii
224
225 return_tolower:
226     move v0, a0                 #Pone el resultado en v0
227     lw gp, 20($fp)
228     lw $fp, 16($fp)
229     addiu sp, sp, 24
230     jr ra
231
232 .end my_tolower
```

```
233 .size my_tolower,.-my_tolower
234
235
236 #
    ------------------------------------------------------------------------
237
238
239
240 .ent leer_palabra
241
242 leer_palabra:
243     .frame $fp, 40, ra
244     .set noreorder
245     .cpload t9
246     .set reorder
247
248     subu sp, sp, 40
249     sw s0, 24(sp)
250     sw $fp, 28(sp)
251     .cprestore 32
252     sw ra, 36(sp)
253     move $fp, sp
254
255     sw a0, 40($fp)          # Guardamos el primer argumento en el
           stackframe (puntero al archivo)
256     sw a1, 44($fp)          # Guardamos el segundo argumento en el
           stackframe (puntero a longitud)
257     sw a2, 48($fp)          # Guardamos el tercer argumento en el
           stackframe (puntero a buffer)
258
259     la t0, TAM
260     lw a0, 0(t0)                        #Carga TAM en a0
261     la t9, mymalloc
262     jalr t9                             #Llama a malloc
263     beq v0, zero, terminar_con_error    #Si malloc devuelve 0 ocurrio un
           error
264
265     sw v0, 16($fp)                      #Guardamos el puntero a la palabra
           en el stackframe
266     addu s0, zero, zero                 # Inicializamos la longitud de la
           palabra en cero
267
268
269 loop_leer_palabra:
270     lw a0, 40($fp)                      # Recuperamos el puntero al archivo
271     lw a1, 48($fp)                      # Recuperamos el puntero al buffer
272     la t9, getch
273     jalr t9                             #Leemos un caracter, queda en v0
274     beq v0, -1, error_leer_palabra      #Si devuelve -1 ocurrio un error
275
276
277     beq v0, ASCII_GUION, es_caracter    # Salta si es -
```

```
278
279     beq v0, ASCII_GUIONBAJO, es_caracter   # Salta si es _
280
281     sgeu t0, v0, ASCII_A_MAY            # Mayor que "A"
282     sleu t1, v0, ASCII_Z_MAY            # Menor que "Z"
283     beq t0, t1, es_caracter             # Salta si es letra mayuscula
284
285     sgeu t0, v0, ASCII_A_MIN            # Mayor que "a"
286     sleu t1, v0, ASCII_Z_MIN            # Menor que "z"
287     beq t0, t1, es_caracter             # Salta si es letra minuscula
288
289     sgeu t0, v0, ASCII_CERO             # Mayor que "0"
290     sleu t1, v0, ASCII_NUEVE            # Menor que "9"
291     beq t0, t1, es_caracter             #Salta si es un numero
292
293 no_es_caracter:
294     lw a0, 16($fp)                      # Recuperamos puntero a palabra
295     addu t0, s0, a0                     #t0 = palabra + len
296     addiu t1, zero, ASCII_NEWLINE
297     sb t1, 0(t0)                        #Guardamos el "\n"
298
299     lw a1, 44($fp)                      # Cargamos puntero a len
300     sw s0, 0(a1)                        # Guardamos el len
301     b terminar_leer_palabra             #Sale del loop
302
303 es_caracter:
304     lw a0, 16($fp)                      # Recuperamos puntero a palabra
305     addu t0, s0, a0                     #t0 = palabra + len
306     sb v0, 0(t0)                        #Guardamos el caracter
307
308     addiu s0, s0, 1                     #Incrementamos len
309
310     la t0, TAM
311     lw t1, 0(t0)                        #Carga TAM en t1
312     remu t2, s0, t1                     #t2 = len %tam
313
314     bne t2, zero, loop_leer_palabra     #Continua al loop si el modulo no
            es 0
315
316     lw a0, 16($fp)                      # Recuperamos puntero a palabra en
            a0
317     move a1, s0                         #a1 = len
318     la t0, TAM
319     lw a2, 0(t0)                        #Carga TAM en a2
320
321     la t9, myrealloc
322     jalr t9                             #Llama a realloc
323     beq v0, zero, error_leer_palabra    #Si realloc devuelve 0 ocurrio un
            error
324     sw v0, 16($fp)                      #Guardamos el nuevo puntero en el
            stackframe
325
326     b loop_leer_palabra                 #Vuelve siempre al loop
```

```
327
328 terminar_leer_palabra:
329     lw v0, 16($fp)                      #Carga en v0 el puntero a la
            palabra
330 return_leer_palabra:
331     lw s0, 24($fp)
332     lw ra, 36($fp)
333     lw gp, 32($fp)
334     lw $fp, 28($fp)
335     addiu sp, sp, 40
336     jr ra
337
338 error_leer_palabra:
339     lw a0, 16($fp)                      # Recuperamos puntero a palabra en
            a0
340     la t9, myfree
341     jalr t9                             #Llama a free con la palabra
342 terminar_con_error:
343     move v0, zero                       #Devuelve 0 si ocurrio un error
344     b return_leer_palabra
345
346 .end leer_palabra
347 .size leer_palabra,.-leer_palabra
348
349
350 #
    ------------------------------------------------------------------------
351
352 .ent crear_buffer
353
354 crear_buffer:
355     .frame $fp, 32, ra
356     .set noreorder
357     .cpload t9
358     .set reorder
359
360     subu sp, sp, 32
361     sw $fp, 16(sp)
362     .cprestore 20
363     sw ra, 24(sp)
364     move $fp, sp
365
366     sw a0, 32($fp)                      # Guardamos el tamanio del buffer
            en el stackframe
367
368     la t9, mymalloc
369     jalr t9                             #Llama a malloc
370
371     bne v0, zero, return_crear_buffer   #Salta si no ocurrio un error
372     move v0, zero                       #Devuelve 0 si ocurrio un error
373
374 return_crear_buffer:
```

```
375     lw ra, 24($fp)
376     lw gp, 20($fp)
377     lw $fp, 16($fp)
378     addiu sp, sp, 32
379     jr ra
380
381 .end crear_buffer
382 .size crear_buffer,.-crear_buffer
383
384
385 #
    ------------------------------------------------------------------------------
386
387
388 .ent getch
389
390 getch:
391
392     .frame $fp, 40, ra
393     .set noreorder
394     .cpload t9
395     .set reorder
396
397     subu sp, sp, 40
398     sw s0, 16(sp)
399     sw s1, 20(sp)
400     sw $fp, 24(sp)
401     .cprestore 28
402     sw ra, 32(sp)
403     move $fp, sp
404
405     sw a0, 40($fp)                  # Guardamos el archivo de entrada
        en el stackframe
406     sw a1, 44($fp)                  # Guardamos el puntero al buffer en
         el stackframe
407
408     la t0, pos_buffer_entrada
409     lw s0, 0(t0)                   #s0 = pos actual del buffer
410
411     la t0, tam_buffer_entrada
412     lw a2, 0(t0)                   #a2 = tam actual del buffer
413
414     bltu s0, a2, leer_caracter      #Salta si pos actual < tam buffer
415
416 #Si no salta tengo que volver a leer del archivo y llenar el buffer
417
418     li v0, SYS_read
419     syscall
420     bne a3, zero, error_getch       #Salta si hubo un error
421     beq v0, zero, leyo_eof_getch    #Si read devuelve 0 leyo EOF
422
423     addu s0, zero, zero             #Pos_actual = 0
```

```
424     move s1,v0                      #s1 = cantidad bytes leidos
425
426     la t0, tam_buffer_entrada
427     lw a2, 0(t0)                    #a2 = tam actual del buffer
428     beq v0,a2, leer_caracter        #Salta si read no leyo menos bytes
           de lo indicado
429
430 #Si leyo menos bytes
431 leer_archivo:
432     lw a0, 40($fp)                  # a0 = archivo de entrada
433     lw a1, 44($fp)
434     addu a1, a1, s1                 #a1 = puntero a buffer + cant bytes
            leidos
435     la t0, tam_buffer_entrada
436     lw a2, 0(t0)
437     subu a2, a2, s1                 #a2 = tam actual del buffer - cant
           bytes leidos
438     li v0, SYS_read
439     syscall
440     bne a3, zero, error_getch       #Salta si hubo un error
441     beq v0, zero, eof_leido_getch   #Si es cero leyo eof
442     addu s1, s1, v0                 #s1 = cant bytes leidos
443     la t0, tam_buffer_entrada
444     lw t1, 0(t0)                    #t1 = tam buffer entrada
445     blt s1, t1, leer_archivo        #Si cant bytes leidos < tam vuelve
           a leer
446     beq s1, t1, leer_caracter       #Si ya leyo todo salta a leer
           caracter
447
448 eof_leido_getch:
449     la t0, tam_buffer_entrada       #Si leyo menos bytes pero todavia
           hay cosas en el buffer
450     sw s1, 0(t0)                    #Actualizo el tamanio del buffer
451
452 leer_caracter:
453
454     lw a1, 44($fp)                  #Cargo el puntero al buffer
455     addu t2, s0, a1                 #t2 = buffer + pos
456     lbu v0, 0(t2)                   #v0 = caracter leido
457     addiu s0, s0, 1                 #Incremento la posicion actual
458
459     la t0, pos_buffer_entrada
460     sw s0, 0(t0)                  #Guardamos la pos actual del buffer
461
462 return_getch:
463     lw s0, 16($fp)
464     lw s1, 20($fp)
465     lw ra, 32($fp)
466     lw gp, 28($fp)
467     lw $fp, 24($fp)
468     addiu sp, sp, 40
469     jr ra
470
```

```
471 leyo_eof_getch:
472     la t0, eof_leido
473     addiu t1,zero,1
474     sw t1, 0(t0)                    #Actualizo la variable EOF leido
475     addu v0, zero, zero             #Devuelvo 0
476     b return_getch
477
478 error_getch:
479     addiu v0, zero, -1              #Devuelvo -1
480     b return_getch
481
482 .end getch
483 .size getch,.-getch
484
485 #
    ------------------------------------------------------------------------------
486
487
488 .ent putch
489
490 putch:
491
492     .frame $fp, 48, ra
493     .set noreorder
494     .cpload t9
495     .set reorder
496
497     subu sp, sp, 48
498     sw s0, 16(sp)
499     sw s1, 20(sp)
500     sw s2, 24(sp)
501     sw s3, 28(sp)
502     sw $fp, 32(sp)
503     .cprestore 36
504     sw ra, 40(sp)
505     move $fp, sp
506
507     sw a0, 48($fp)                  # Guardamos el archivo de salida en
          el stackframe
508     sw a1, 52($fp)                  # Guardamos el puntero a la palabra
          en el stackframe
509     sw a2, 56($fp)                  # Guardamos el buffer en el
        stackframe
510
511     la t0, pos_buffer_salida
512     lw s0, 0(t0)                  #s0 = pos actual del buffer
513
514     addu s1, zero, zero             #Inicializamos el indice de la
        palabra en s1
515
516     la t0, eof_escrito
517     lw t1, 0(t0)                  #t1 = eof debe ser escrito?
```

```
518     beq t1,zero, loop_putch             #Salta si eof no debe ser escrito
519
520     move s3, zero                       #s3 = cant bytes escritos = 0
521     la t0, tam_buffer_salida
522     sw s0, 0(t0)                      #Actualiza el tamanio del buffer a la
            posicion actual
523     b escribir_todo_putch
524
525
526 loop_putch:
527     lw a1, 52($fp)                      #Cargamos el puntero a la palabra
528     addu t3, a1, s1                     #t3 = palabra + indice
529     lbu s2, 0(t3)                       #s2 = caracter a escribir
530
531     lw a2, 56($fp)                      #Cargamos el puntero al buffer
532     addu t5, a2, s0                     #t5 = buffer + pos_actual
533     sb s2, 0(t5)                        #Guardamos el caracter
534
535     addiu s0,s0,1                       #Incrementamos pos_actual
536     la t0, tam_buffer_salida
537     lw t1, 0(t0)                        #Cargamos el tamanio del buffer
538     bltu s0, t1, continuar_loop_putch   #Salta si pos_actual < tam_buffer
539     move s3, zero                       #s3 = cant bytes escritos = 0
540
541 #Si no salta tengo que volver a escribir el archivo y vaciar el buffer
542
543 escribir_todo_putch:
544     lw a0, 48($fp)                      # Cargamos el archivo de salida
545     lw a1, 56($fp)                      # Cargamos el buffer
546     addu a1, a1, s3                     #a1 = buffer + cant bytes escritos
547     la t0, tam_buffer_salida
548     lw a2, 0(t0)                        # Cargamos el tamanio total del
            buffer
549     subu a2, a2, s3                     #a2 = tam - cant bytes escritos
550     li v0, SYS_write
551     syscall
552
553     bne a3, zero, error_putch           #Salta si ocurrio un error
554     addu s3, s3, v0                     #s3 = cant bytes escritos
555     la t0, tam_buffer_salida
556     lw t1, 0(t0)                        # t1 = tamanio total del buffer
557     blt s3, t1, escribir_todo_putch     #Si escribio menos bytes vuelve a
            escribir
558
559     addu s0, zero, zero                 #Pos_actual = 0
560
561     la t0, eof_escrito
562     lw t1, 0(t0)                      #t1 = eof_escrito
563     bgt t1, zero, terminar_putch        #Termina si EOF fue escrito
564
565 continuar_loop_putch:
566     addu s1, s1, 1                            #Incrementamos el indice de
            la palabra
```

```
567     beq s2, ASCII_NEWLINE, terminar_putch       #Termina si es \n
568     b loop_putch                                #Vuelve al loop
569
570 terminar_putch:
571     addiu v0, zero, 1                   #Devuelve 1 si no ocurrio un error
572 return_putch:
573     la t0, pos_buffer_salida
574     sw s0, 0(t0)
575     lw s0, 16($fp)
576     lw s1, 20($fp)
577     lw s2, 24($fp)
578     lw s3, 28($fp)
579     lw ra, 40($fp)
580     lw gp, 36($fp)
581     lw $fp, 32($fp)
582     addiu sp, sp, 48
583     jr ra
584
585 error_putch:
586     move v0, zero                       #Devuelve 0 si ocurrio un error
587     b return_putch
588
589 .end putch
590 .size putch,.-putch
591
592 #-------------------------------------------------
593
594 .data
595
596 pos_buffer_entrada: .word -1
597 tam_buffer_entrada: .word 0
598 pos_buffer_salida: .word 0
599 tam_buffer_salida: .word 0
600 eof_leido: .word 0
601 eof_escrito: .word 0
602 TAM: .word 50
```