

Trabajo Práctico Nro. 1: programación MIPS

Lucas Verón, *Padrón Nro. 89.341*
lucasveron86@gmail.com

Eliana Diaz, *Padrón Nro. 89.324*
diazeliana09@gmail.com

Alan Helouani, *Padrón Nro. 90.289*
alanhelouani@gmail.com

2do. Cuatrimestre de 2017
66.20 Organización de Computadoras – Práctica Martes
Facultad de Ingeniería, Universidad de Buenos Aires

Resumen

El presente proyecto tiene por finalidad familiarizarnos con el conjunto de instrucciones MIPS y el concepto de ABI

1. Introducción

Se detallará el diseño e implementación de un programa en lenguaje C y MIPS que procesa archivos de texto por línea de comando, como así también la forma de ejecución del mismo y los resultados obtenidos en las distintas pruebas ejecutadas.

El programa recibe los archivos o streams de entrada y salida, e imprime aquellas palabras del archivo de entrada (componentes léxicos) que sean palíndromos.

Se define como *palabra* a aquellos componentes léxicos del stream de entrada compuestos exclusivamente por combinaciones de caracteres a-z, 0-9, - (signo menos) y (*guinbajo*).

Por otro lado, se considera que una palabra, número o frase, es *palíndroma* cuando se lee igual hacía adelante que hacía atrás.

Se implementará una función "palindrome" la cual se encargará de verificar si efectivamente la palabra es o no palíndroma. La función estará escrita en assembly MIPS.

Los streams serán leídos y escritos de a bloques de memoria configurables, los cuales serán almacenados en un "buffer" para luego ser leídos de a uno.

2. Diseño

Las funcionalidades requeridas son las siguientes:

- Ayuda (Help): Presentación un detalle de los comandos que se pueden ejecutar.

- Versión: Se debe indicar la versión del programa.
- Procesar los datos:
 - Con especificación sólo del archivo de entrada.
 - Con especificación sólo del archivo de salida.
 - Con especificación del archivo de entrada y de salida.
 - Sin especificación del archivo de entrada ni de salida.
- Setting del tamaño del buffer in y buffer out; indicando de a cuantos caracteres se debe leer y escribir.

En base a estas funcionalidades, se modularizó el código a fin de poder reutilizarlo y a su vez que cada método se encargue de ejecutar una única funcionalidad.

3. Implementación

3.1. Código fuente en lenguaje C

```

1  /*
2  =====
3
4  Name      : tp1.c
5  Author    : Grupo orga 66.20
6  Version   : 1
7  Copyright : Orga6620 - Tp1
8  Description : Trabajo practico 1: Programacion MIPS
9  =====
10
11 */
12
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <string.h>
16 #include <getopt.h>
17 #include <unistd.h>
18 #include "process.h"
19
20 #define VERSION "1.0"
21
22 #define FALSE 0
23 #define TRUE 1
24
25 size_t ibytes = 1;
26 size_t obytes = 1;
27
28 enum ParameterState {
29     OKEY = 0, INCORRECT_QUANTITY_PARAMS = 1,
30     INCORRECT_MENU = 2, ERROR_FILE = 3, ERROR_MEMORY =
31     4, ERROR_READ = 5, ERROR_BYTES = 6
32 };

```

```

29
30 int executeHelp() {
31     fprintf(stdout, "Usage: \n");
32     fprintf(stdout, "        tpl -h \n");
33     fprintf(stdout, "        tpl -V \n");
34     fprintf(stdout, "        tpl [options] \n");
35     fprintf(stdout, "Options: \n");
36     fprintf(stdout, "        -V, --version          Print
        version and quit. \n");
37     fprintf(stdout, "        -h, --help            Print
        this information. \n");
38     fprintf(stdout, "        -i, --input            Location of the input file. \n");
39     fprintf(stdout, "        -o, --output          Location of the output file. \n");
40     fprintf(stdout, "        -I, --ibuf-bytes      Byte-
        count of the input buffer. \n");
41     fprintf(stdout, "        -O, --obuf-bytes      Byte-
        count of the output buffer. \n");
42     fprintf(stdout, "Examples: \n");
43     fprintf(stdout, "        tpl -i ~/input -o ~/output \n")
44     ;
45     return OKEY;
46 }
47
48 int executeVersion() {
49     fprintf(stdout, "Version: \"%s\" \n", VERSION);
50
51     return OKEY;
52 }
53
54 int executeWithDefaultParameter(char * path, int isInputDefault
55 , int isOutputDefault) {
56     FILE * fileInput = NULL;
57     FILE * fileOutput = NULL;
58
59     if (isInputDefault == TRUE && isOutputDefault == TRUE)
60     {
61         fileInput = stdin;
62         fileOutput = stdout;
63     } else {
64         if (isInputDefault == TRUE) {
65             fileInput = stdin;
66
67             fileOutput = fopen(path, "w"); // Opens
68             a text file for writing. Place the
69             content.
70             if (fileOutput == NULL) {
71                 fprintf(stderr, "[Error] El
72                 archivo de output no pudo
73                 ser abierto para escritura:
74                 %s \n", path);
75                 return ERROR_FILE;
76             }
77         }
78     }

```

```

69         }
70     } else {
71         fileInput = fopen(path, "r"); // Opens
                                         an existing text file for reading
                                         purpose.
72         if (fileInput == NULL) {
73             fprintf(stderr, "[Error] El
                                         archivo de input no pudo ser
                                         abierto para lectura: %s \n
                                         ", path);
74             return ERROR_FILE;
75         }
76
77         fileOutput = stdout;
78     }
79 }
80
81 int ifd = fileno(fileInput);
82 int ofd = fileno(fileOutput);
83
84 int executeResult = palindrome(ifd, ibytes, ofd, obytes
85 );
86
87 if (isInputDefault == FALSE || isOutputDefault == FALSE
88 ) {
89     if (isInputDefault == TRUE) {
90         if (fileOutput != NULL) {
91             int result = fclose(fileOutput)
92             ;
93             if (result == EOF) {
94                 fprintf(stderr, "[
95                     Warning] El archivo
96                     de output no pudo
97                     ser cerrado
98                     correctamente: %s \n
99                     ", path);
100                 return ERROR_FILE;
101             }
102         }
103     } else {
104         if (fileInput != NULL) {
105             int result = fclose(fileInput);
106             if (result == EOF) {
107                 fprintf(stderr, "[
108                     Warning] El archivo
109                     de input no pudo ser
110                     cerrado
111                     correctamente: %s \n
112                     ", path);
113                 return ERROR_FILE;
114             }
115         }
116     }
117 }

```

```

105         return executeResult;
106     }
107 }
108
109 int executeWithParameters(char * pathInput, char * pathOutput)
110 {
111     FILE * fileInput = fopen(pathInput, "r"); // Opens an
112         existing text file for reading purpose.
113     if (fileInput == NULL) {
114         fprintf(stderr, "[Error] El archivo de input no
115             pudo ser abierto para lectura: %s \n",
116             pathInput);
117         return ERROR_FILE;
118     }
119
120     FILE * fileOutput = fopen(pathOutput, "w"); // Opens a
121         text file for writing. Place the content.
122     if (fileOutput == NULL) {
123         fprintf(stderr, "[Error] El archivo de output
124             no pudo ser abierto para escritura: %s \n",
125             pathOutput);
126
127         int result = fclose(fileInput);
128         if (result == EOF) {
129             fprintf(stderr, "[Warning] El archivo
130                 de input no pudo ser cerrado
131                 correctamente: %s \n", pathInput);
132         }
133         return ERROR_FILE;
134     }
135
136     int ifd = fileno(fileInput);
137     int ofd = fileno(fileOutput);
138
139     int executeResult = palindrome(ifd, ibytes, ofd, obytes
140         );
141
142     int resultFileInputClose = 0; // EOF = -1
143     if (fileInput != NULL) {
144         resultFileInputClose = fclose(fileInput);
145         if (resultFileInputClose == EOF) {
146             fprintf(stderr, "[Warning] El archivo
147                 de input no pudo ser cerrado
148                 correctamente: %s \n", pathInput);
149         }
150     }
151
152     if (fileOutput != NULL) {
153         int result = fclose(fileOutput);
154         if (result == EOF) {
155             fprintf(stderr, "[Warning] El archivo
156                 de output no pudo ser cerrado
157                 correctamente: %s \n", pathOutput);
158         }
159     }
160 }

```

```

145         return ERROR_FILE;
146     }
147 }
148
149 if (resultFileInputClose) {
150     return ERROR_FILE;
151 }
152
153 return executeResult;
154 }
155
156 int executeByMenu(int argc, char **argv) {
157     // Always begins with /
158     if (argc == 1) {
159         // Run with default parameters
160         return executeWithDefaultParameter(NULL, TRUE,
161             TRUE);
162     }
163
164     char * inputValue = NULL;
165     char * outputValue = NULL;
166     char * iBufBytes = NULL;
167     char * oBufBytes = NULL;
168
169     /* Una cadena que lista las opciones cortas validas */
170     const char* const smallOptions = "Vhi:o:I:O:";
171
172     /* Una estructura de varios arrays describiendo los
173        valores largos */
174     const struct option longOptions[] = {
175         {"version",          no_argument,
176             0, 'V' },
177         {"help",            no_argument,
178             0, 'h' },
179         {"input",           required_argument, 0,
180             'i' }, // optional_argument
181         {"output",         required_argument, 0,
182             'o' },
183         {"ibuf-bytes",     required_argument, 0, 'I'
184             },
185         {"obuf-bytes",     required_argument, 0, 'O'
186             },
187         {0,                0,
188             0, 0 }
189     };
190
191     int incorrectOption = FALSE;
192     int finish = FALSE;
193     int result = OKEY;
194     int longIndex = 0;
195     char opt = 0;
196
197     while ((opt = getopt_long(argc, argv, smallOptions,

```

```

189                                     longOptions, &longIndex ))
                                     != -1 && incorrectOption
                                     == FALSE && finish ==
                                     FALSE) {
190     switch (opt) {
191         case 'V' :
192             result = executeVersion();
193             finish = TRUE;
194             break;
195         case 'h' :
196             result = executeHelp();
197             finish = TRUE;
198             break;
199         case 'i' :
200             inputValue = optarg;
201             break;
202         case 'o' :
203             outputValue = optarg;
204             break;
205         case 'I' :
206             iBufBytes = optarg;
207             break;
208         case 'O' :
209             oBufBytes = optarg;
210             break;
211         default:
212             incorrectOption = TRUE;
213     }
214 }
215
216 if (incorrectOption == TRUE) {
217     fprintf(stderr, "[Error] Incorrecta option de
218         menu.\n");
219     return INCORRECT_MENU;
220 }
221
222 if (finish == TRUE) {
223     return result;
224 }
225
226 if (iBufBytes != NULL) {
227     char *finalPtr;
228     abytes = strtoul(iBufBytes, &finalPtr, 10);
229     if (abytes == 0) {
230         fprintf(stderr, "[Error] Incorrecta
231             cantidad de bytes para el buffer de
232             entrada.\n");
233         return ERROR_BYTES;
234     }
235 }
236
237 if (oBufBytes != NULL) {
238     char *finalPtr;
239     obytes = strtoul(oBufBytes, &finalPtr, 10);

```

```

237         if (obytes == 0) {
238             fprintf(stderr, "[Error] Incorrecta
                cantidad de bytes para el buffer de
                salida.\n");
239             return ERROR_BYTES;
240         }
241     }
242
243     if (inputValue == NULL && outputValue == NULL) {
244         return executeWithDefaultParameter(NULL, TRUE,
            TRUE);
245     }
246
247     // / -i fileInput
248     if (inputValue != NULL && outputValue == NULL) {
249         if (strcmp("-",inputValue) == 0) {
250             return executeWithDefaultParameter(NULL
                , TRUE, TRUE);
251         } else {
252             return executeWithDefaultParameter(
                inputValue, FALSE, TRUE);
253         }
254     }
255
256     // / -o fileOutput
257     if (inputValue == NULL && outputValue != NULL) {
258         if (strcmp("-",outputValue) == 0) {
259             return executeWithDefaultParameter(NULL
                , TRUE, TRUE);
260         } else {
261             return executeWithDefaultParameter(
                outputValue, TRUE, FALSE);
262         }
263     }
264
265     if (inputValue != NULL && outputValue != NULL) {
266         if (strcmp("-",inputValue) == 0 && strcmp("-",
            outputValue) == 0) {
267             return executeWithDefaultParameter(NULL
                , TRUE, TRUE);
268         }
269
270         if (strcmp("-",inputValue) == 0 && strcmp("-",
            outputValue) != 0) {
271             return executeWithDefaultParameter(
                outputValue, TRUE, FALSE);
272         }
273
274         if (strcmp("-",inputValue) != 0 && strcmp("-",
            outputValue) == 0) {
275             return executeWithDefaultParameter(
                inputValue, FALSE, TRUE);
276         }
277

```



```

278         return executeWithParameters(inputValue,
279                                     outputValue);
280     }
281     fprintf(stderr, "[Error] Incorrecta option de menu.\n");
282     ;
283     return INCORRECT_MENU;
284 }
285
286 int main(int argc, char **argv) {
287     // / -i lalala.txt -o pepe.txt -I 2 -O 3 => 9
288     // parameters as maximum
289     if (argc > 9) {
290         fprintf(stderr, "[Error] Cantidad máxima de
291                 parámetros incorrecta: %d \n", argc);
292         return INCORRECT_QUANTITY_PARAMS;
293     }
294     return executeByMenu(argc, argv);
295 }

```

3.2. Código MIPS32

```

1      .file    1 "tp1.c"
2      .section .mdebug.abi32
3      .previous
4      .abicalls
5      .globl   ibytes
6      .data
7      .align   2
8      .type    ibytes, @object
9      .size    ibytes, 4
10     ibytes:
11         .word    1
12         .globl   obytes
13         .align   2
14         .type    obytes, @object
15         .size    obytes, 4
16     obytes:
17         .word    1
18         .rdata
19         .align   2
20     $LC0:
21         .ascii   "Usage: \n\000"
22         .align   2
23     $LC1:
24         .ascii   "\ttp1 -h \n\000"
25         .align   2
26     $LC2:
27         .ascii   "\ttp1 -V \n\000"
28         .align   2
29     $LC3:
```

```

30      .ascii  "\ttp1 [options] \n\000"
31      .align  2
32 $LC4:
33      .ascii  "Options: \n\000"
34      .align  2
35 $LC5:
36      .ascii  "\t-V, --version\t\tPrint version and quit. \n
37              \000"
38      .align  2
39 $LC6:
40      .ascii  "\t-h, --help\t\t\tPrint this information. \n
41              \000"
42      .align  2
43 $LC7:
44      .ascii  "\t-i, --input\t\t\tLocation of the input
45              file. \n\000"
46      .align  2
47 $LC8:
48      .ascii  "\t-o, --output\t\tLocation of the output file.
49              \n\000"
50      .align  2
51 $LC9:
52      .ascii  "\t-I, --ibuf-bytes\tByte-count of the input
53              buffer. \n\000"
54      .align  2
55 $LC10:
56      .ascii  "\t-O, --obuf-bytes\tByte-count of the output
57              buffer. \n\000"
58      .align  2
59 $LC11:
60      .ascii  "Examples: \n\000"
61      .align  2
62 $LC12:
63      .ascii  "\ttp1 -i ~/input -o ~/output \n\000"
64      .text
65      .align  2
66      .globl  executeHelp
67      .ent    executeHelp
68 executeHelp:
69      .frame  $fp,40,$ra          # vars= 0, regs= 3/0,
70          args= 16, extra= 8
71      .mask   0xd0000000,-8
72      .fmask  0x00000000,0
73      .set    noreorder
74      .cpld   $t9
75      .set    reorder
76      subu    $sp,$sp,40
77      .cprestore 16
78      sw      $ra,32($sp)
79      sw      $fp,28($sp)
80      sw      $gp,24($sp)
81      move    $fp,$sp
82      la      $a0,___sF+88
83      la      $a1,$LC0

```

```

77      la      $t9, fprintf
78      jal     $ra, $t9
79      la      $a0, __sF+88
80      la      $a1, $LC1
81      la      $t9, fprintf
82      jal     $ra, $t9
83      la      $a0, __sF+88
84      la      $a1, $LC2
85      la      $t9, fprintf
86      jal     $ra, $t9
87      la      $a0, __sF+88
88      la      $a1, $LC3
89      la      $t9, fprintf
90      jal     $ra, $t9
91      la      $a0, __sF+88
92      la      $a1, $LC4
93      la      $t9, fprintf
94      jal     $ra, $t9
95      la      $a0, __sF+88
96      la      $a1, $LC5
97      la      $t9, fprintf
98      jal     $ra, $t9
99      la      $a0, __sF+88
100     la      $a1, $LC6
101     la      $t9, fprintf
102     jal     $ra, $t9
103     la      $a0, __sF+88
104     la      $a1, $LC7
105     la      $t9, fprintf
106     jal     $ra, $t9
107     la      $a0, __sF+88
108     la      $a1, $LC8
109     la      $t9, fprintf
110     jal     $ra, $t9
111     la      $a0, __sF+88
112     la      $a1, $LC9
113     la      $t9, fprintf
114     jal     $ra, $t9
115     la      $a0, __sF+88
116     la      $a1, $LC10
117     la      $t9, fprintf
118     jal     $ra, $t9
119     la      $a0, __sF+88
120     la      $a1, $LC11
121     la      $t9, fprintf
122     jal     $ra, $t9
123     la      $a0, __sF+88
124     la      $a1, $LC12
125     la      $t9, fprintf
126     jal     $ra, $t9
127     move     $v0, $zero
128     move     $sp, $fp
129     lw      $ra, 32($sp)
130     lw      $fp, 28($sp)

```

```

131         addu    $sp,$sp,40
132         j        $ra
133     .end        executeHelp
134     .size       executeHelp, .-executeHelp
135     .rdata
136     .align      2
137 $LC13:
138     .ascii      "Version: \"%s\" \n\000"
139     .align      2
140 $LC14:
141     .ascii      "1.0\000"
142     .text
143     .align      2
144     .globl      executeVersion
145     .ent        executeVersion
146 executeVersion:
147     .frame      $fp,40,$ra          # vars= 0, regs= 3/0,
148         args= 16, extra= 8
149     .mask       0xd0000000,-8
150     .fmask      0x00000000,0
151     .set        noreorder
152     .cpload     $t9
153     .set        reorder
154     subu        $sp,$sp,40
155     .cpstore    16
156     sw          $ra,32($sp)
157     sw          $fp,28($sp)
158     sw          $gp,24($sp)
159     move        $fp,$sp
160     la          $a0,___sf+88
161     la          $a1,$LC13
162     la          $a2,$LC14
163     la          $t9,fprintf
164     jal         $ra,$t9
165     move        $v0,$zero
166     move        $sp,$fp
167     lw          $ra,32($sp)
168     lw          $fp,28($sp)
169     addu        $sp,$sp,40
170     j          $ra
171     .end        executeVersion
172     .size       executeVersion, .-executeVersion
173     .rdata
174     .align      2
175 $LC15:
176     .ascii      "w\000"
177     .align      2
178 $LC16:
179     .ascii      "[Error] El archivo de output no pudo ser
180         abierto para es"
181     .ascii      "critura: %s \n\000"
182     .align      2
183 $LC17:
184     .ascii      "r\000"

```

```

183     .align 2
184 $LC18:
185     .ascii "[Error] El archivo de input no pudo ser
        abierto para lec"
186     .ascii "tura: %s \n\000"
187     .align 2
188 $LC19:
189     .ascii "[Warning] El archivo de output no pudo ser
        cerrado corre"
190     .ascii "ctamente: %s \n\000"
191     .align 2
192 $LC20:
193     .ascii "[Warning] El archivo de input no pudo ser
        cerrado correc"
194     .ascii "tamente: %s \n\000"
195     .text
196     .align 2
197     .globl executeWithDefaultParameter
198     .ent executeWithDefaultParameter
199 executeWithDefaultParameter:
200     .frame $fp,72,$ra # vars= 32, regs= 3/0,
        args= 16, extra= 8
201     .mask 0xd0000000,-8
202     .fmask 0x00000000,0
203     .set noreorder
204     .cpload $t9
205     .set reorder
206     subu $sp,$sp,72
207     .cpstore 16
208     sw $ra,64($sp)
209     sw $fp,60($sp)
210     sw $gp,56($sp)
211     move $fp,$sp
212     sw $a0,72($fp)
213     sw $a1,76($fp)
214     sw $a2,80($fp)
215     sw $zero,24($fp)
216     sw $zero,28($fp)
217     lw $v1,76($fp)
218     li $v0,1 # 0x1
219     bne $v1,$v0,$L20
220     lw $v1,80($fp)
221     li $v0,1 # 0x1
222     bne $v1,$v0,$L20
223     la $v0, __sF
224     sw $v0,24($fp)
225     la $v0, __sF+88
226     sw $v0,28($fp)
227     b $L21
228 $L20:
229     lw $v1,76($fp)
230     li $v0,1 # 0x1
231     bne $v1,$v0,$L22
232     la $v0, __sF

```

```

233         sw      $v0,24($fp)
234         lw      $a0,72($fp)
235         la      $a1,$LC15
236         la      $t9,fopen
237         jal     $ra,$t9
238         sw      $v0,28($fp)
239         lw      $v0,28($fp)
240         bne     $v0,$zero,$L21
241         la      $a0,___sF+176
242         la      $a1,$LC16
243         lw      $a2,72($fp)
244         la      $t9,fprintf
245         jal     $ra,$t9
246         li      $v0,3                      # 0x3
247         sw      $v0,48($fp)
248         b       $L19
249 $L22:
250         lw      $a0,72($fp)
251         la      $a1,$LC17
252         la      $t9,fopen
253         jal     $ra,$t9
254         sw      $v0,24($fp)
255         lw      $v0,24($fp)
256         bne     $v0,$zero,$L25
257         la      $a0,___sF+176
258         la      $a1,$LC18
259         lw      $a2,72($fp)
260         la      $t9,fprintf
261         jal     $ra,$t9
262         li      $v0,3                      # 0x3
263         sw      $v0,48($fp)
264         b       $L19
265 $L25:
266         la      $v0,___sF+88
267         sw      $v0,28($fp)
268 $L21:
269         lw      $v0,24($fp)
270         lh      $v0,14($v0)
271         sw      $v0,32($fp)
272         lw      $v0,28($fp)
273         lh      $v0,14($v0)
274         sw      $v0,36($fp)
275         lw      $a0,32($fp)
276         lw      $a1,ibytes
277         lw      $a2,36($fp)
278         lw      $a3,obytes
279         la      $t9,palindrome
280         jal     $ra,$t9
281         sw      $v0,40($fp)
282         lw      $v0,76($fp)
283         beq     $v0,$zero,$L27
284         lw      $v0,80($fp)
285         bne     $v0,$zero,$L26
286 $L27:

```

```

287         lw      $v1,76($fp)
288         li      $v0,1                      # 0x1
289         bne     $v1,$v0,$L28
290         lw      $v0,28($fp)
291         beq     $v0,$zero,$L26
292         lw      $a0,28($fp)
293         la      $t9,fclose
294         jal     $ra,$t9
295         sw      $v0,44($fp)
296         lw      $v1,44($fp)
297         li      $v0,-1                     # 0xffffffffffffffff
298         bne     $v1,$v0,$L26
299         la      $a0,___sF+176
300         la      $a1,$LC19
301         lw      $a2,72($fp)
302         la      $t9,fprintf
303         jal     $ra,$t9
304         li      $v0,3                      # 0x3
305         sw      $v0,48($fp)
306         b       $L19
307 $L28:
308         lw      $v0,24($fp)
309         beq     $v0,$zero,$L26
310         lw      $a0,24($fp)
311         la      $t9,fclose
312         jal     $ra,$t9
313         sw      $v0,44($fp)
314         lw      $v1,44($fp)
315         li      $v0,-1                     # 0xffffffffffffffff
316         bne     $v1,$v0,$L26
317         la      $a0,___sF+176
318         la      $a1,$LC20
319         lw      $a2,72($fp)
320         la      $t9,fprintf
321         jal     $ra,$t9
322         li      $v0,3                      # 0x3
323         sw      $v0,48($fp)
324         b       $L19
325 $L26:
326         lw      $v0,40($fp)
327         sw      $v0,48($fp)
328 $L19:
329         lw      $v0,48($fp)
330         move     $sp,$fp
331         lw      $ra,64($sp)
332         lw      $fp,60($sp)
333         addu     $sp,$sp,72
334         j       $ra
335         .end     executeWithDefaultParameter
336         .size    executeWithDefaultParameter, .-
337                 executeWithDefaultParameter
338         .align   2
339         .globl   executeWithParameters
340         .ent     executeWithParameters

```

```

340 executeWithParameters:
341     .frame $fp,72,$ra                # vars= 32, regs= 3/0,
342     args= 16, extra= 8
343     .mask 0xd0000000,-8
344     .fmask 0x00000000,0
345     .set noreorder
346     .cpload $t9
347     .set reorder
348     subu $sp,$sp,72
349     .cpstore 16
350     sw $ra,64($sp)
351     sw $fp,60($sp)
352     sw $gp,56($sp)
353     move $fp,$sp
354     sw $a0,72($fp)
355     sw $a1,76($fp)
356     lw $a0,72($fp)
357     la $a1,$LC17
358     la $t9,fopen
359     jal $ra,$t9
360     sw $v0,24($fp)
361     lw $v0,24($fp)
362     bne $v0,$zero,$L35
363     la $a0,___sF+176
364     la $a1,$LC18
365     lw $a2,72($fp)
366     la $t9,fprintf
367     jal $ra,$t9
368     li $v0,3                        # 0x3
369     sw $v0,52($fp)
370     b $L34
371 $L35:
372     lw $a0,76($fp)
373     la $a1,$LC15
374     la $t9,fopen
375     jal $ra,$t9
376     sw $v0,28($fp)
377     lw $v0,28($fp)
378     bne $v0,$zero,$L36
379     la $a0,___sF+176
380     la $a1,$LC16
381     lw $a2,76($fp)
382     la $t9,fprintf
383     jal $ra,$t9
384     lw $a0,24($fp)
385     la $t9,fclose
386     jal $ra,$t9
387     sw $v0,32($fp)
388     lw $v1,32($fp)
389     li $v0,-1                       # 0xffffffffffffffff
390     bne $v1,$v0,$L37
391     la $a0,___sF+176
392     la $a1,$LC20
393     lw $a2,72($fp)

```



```

393         la      $t9, fprintf
394         jal     $ra, $t9
395 $L37:
396         li      $v0, 3                      # 0x3
397         sw      $v0, 52($fp)
398         b       $L34
399 $L36:
400         lw      $v0, 24($fp)
401         lh      $v0, 14($v0)
402         sw      $v0, 32($fp)
403         lw      $v0, 28($fp)
404         lh      $v0, 14($v0)
405         sw      $v0, 36($fp)
406         lw      $a0, 32($fp)
407         lw      $a1, ibytes
408         lw      $a2, 36($fp)
409         lw      $a3, obytes
410         la      $t9, palindrome
411         jal     $ra, $t9
412         sw      $v0, 40($fp)
413         sw      $zero, 44($fp)
414         lw      $v0, 24($fp)
415         beq     $v0, $zero, $L38
416         lw      $a0, 24($fp)
417         la      $t9, fclose
418         jal     $ra, $t9
419         sw      $v0, 44($fp)
420         lw      $v1, 44($fp)
421         li      $v0, -1                      # 0xffffffffffffffff
422         bne     $v1, $v0, $L38
423         la      $a0, __sF+176
424         la      $a1, $LC20
425         lw      $a2, 72($fp)
426         la      $t9, fprintf
427         jal     $ra, $t9
428 $L38:
429         lw      $v0, 28($fp)
430         beq     $v0, $zero, $L40
431         lw      $a0, 28($fp)
432         la      $t9, fclose
433         jal     $ra, $t9
434         sw      $v0, 48($fp)
435         lw      $v1, 48($fp)
436         li      $v0, -1                      # 0xffffffffffffffff
437         bne     $v1, $v0, $L40
438         la      $a0, __sF+176
439         la      $a1, $LC19
440         lw      $a2, 76($fp)
441         la      $t9, fprintf
442         jal     $ra, $t9
443         li      $v0, 3                      # 0x3
444         sw      $v0, 52($fp)
445         b       $L34
446 $L40:

```

```

447         lw      $v0,44($fp)
448         beq     $v0,$zero,$L42
449         li      $v0,3                      # 0x3
450         sw      $v0,52($fp)
451         b       $L34
452 $L42:
453         lw      $v0,40($fp)
454         sw      $v0,52($fp)
455 $L34:
456         lw      $v0,52($fp)
457         move    $sp,$fp
458         lw      $ra,64($sp)
459         lw      $fp,60($sp)
460         addu    $sp,$sp,72
461         j       $ra
462         .end    executeWithParameters
463         .size   executeWithParameters,.-executeWithParameters
464         .rdata
465         .align  2
466 $LC22:
467         .ascii  "version\000"
468         .align  2
469 $LC23:
470         .ascii  "help\000"
471         .align  2
472 $LC24:
473         .ascii  "input\000"
474         .align  2
475 $LC25:
476         .ascii  "output\000"
477         .align  2
478 $LC26:
479         .ascii  "ibuf-bytes\000"
480         .align  2
481 $LC27:
482         .ascii  "obuf-bytes\000"
483         .data
484         .align  2
485 $LC28:
486         .word   $LC22
487         .word   0
488         .word   0
489         .word   86
490         .word   $LC23
491         .word   0
492         .word   0
493         .word   104
494         .word   $LC24
495         .word   1
496         .word   0
497         .word   105
498         .word   $LC25
499         .word   1
500         .word   0

```

```

501         .word    111
502         .word    $LC26
503         .word    1
504         .word    0
505         .word    73
506         .word    $LC27
507         .word    1
508         .word    0
509         .word    79
510         .word    0
511         .word    0
512         .word    0
513         .word    0
514         .globl   memcpy
515         .rdata
516         .align   2
517 $LC21:
518         .ascii   "Vhi:o:I:O:\000"
519         .align   2
520 $LC29:
521         .ascii   "[Error] Incorrecta option de menu.\n\000"
522         .align   2
523 $LC30:
524         .ascii   "[Error] Incorrecta cantidad de bytes para el
                    buffer de e"
525         .ascii   "ntrada.\n\000"
526         .align   2
527 $LC31:
528         .ascii   "[Error] Incorrecta cantidad de bytes para el
                    buffer de s"
529         .ascii   "alida.\n\000"
530         .align   2
531 $LC32:
532         .ascii   "-\000"
533         .text
534         .align   2
535         .globl   executeByMenu
536         .ent     executeByMenu
537 executeByMenu:
538         .frame   $fp,224,$ra          # vars= 176, regs= 3/0,
                    args= 24, extra= 8
539         .mask    0xd0000000,-8
540         .fmask   0x00000000,0
541         .set     noreorder
542         .cpload  $t9
543         .set     reorder
544         subu     $sp,$sp,224
545         .cprestore 24
546         sw      $ra,216($sp)
547         sw      $fp,212($sp)
548         sw      $gp,208($sp)
549         move    $fp,$sp
550         sw      $a0,224($fp)
551         sw      $a1,228($fp)

```

```

552         lw      $v1,224($fp)
553         li      $v0,1                      # 0x1
554         bne     $v1,$v0,$L44
555         move    $a0,$zero
556         li      $a1,1                      # 0x1
557         li      $a2,1                      # 0x1
558         la      $t9,executeWithDefaultParameter
559         jal     $ra,$t9
560         sw      $v0,196($fp)
561         b       $L43
562 $L44:
563         sw      $zero,32($fp)
564         sw      $zero,36($fp)
565         sw      $zero,40($fp)
566         sw      $zero,44($fp)
567         la      $v0,$LC21
568         sw      $v0,48($fp)
569         addu    $v0,$fp,56
570         la      $v1,$LC28
571         move    $a0,$v0
572         move    $a1,$v1
573         li      $a2,112                    # 0x70
574         la      $t9,memcpy
575         jal     $ra,$t9
576         sw      $zero,168($fp)
577         sw      $zero,172($fp)
578         sw      $zero,176($fp)
579         sw      $zero,180($fp)
580         sb      $zero,184($fp)
581 $L45:
582         addu    $v1,$fp,56
583         addu    $v0,$fp,180
584         sw      $v0,16($sp)
585         lw      $a0,224($fp)
586         lw      $a1,228($fp)
587         lw      $a2,48($fp)
588         move    $a3,$v1
589         la      $t9,getopt_long
590         jal     $ra,$t9
591         sb      $v0,184($fp)
592         lbu     $v0,184($fp)
593         sll     $v0,$v0,24
594         sra     $v1,$v0,24
595         li      $v0,-1                     # 0xffffffffffffffff
596         beq     $v1,$v0,$L46
597         lw      $v0,168($fp)
598         bne     $v0,$zero,$L46
599         lw      $v0,172($fp)
600         bne     $v0,$zero,$L46
601         lb      $v0,184($fp)
602         addu    $v0,$v0,-73
603         sw      $v0,200($fp)
604         lw      $v1,200($fp)
605         sltu    $v0,$v1,39

```

```

606         beq      $v0,$zero,$L56
607         lw       $v0,200($fp)
608         sll      $v1,$v0,2
609         la       $v0,$L57
610         addu     $v0,$v1,$v0
611         lw       $v0,0($v0)
612         .cpadd   $v0
613         j        $v0
614         .rdata
615         .align   2
616 $L57:
617         .gpword  $L54
618         .gpword  $L56
619         .gpword  $L56
620         .gpword  $L56
621         .gpword  $L56
622         .gpword  $L56
623         .gpword  $L55
624         .gpword  $L56
625         .gpword  $L56
626         .gpword  $L56
627         .gpword  $L56
628         .gpword  $L56
629         .gpword  $L56
630         .gpword  $L50
631         .gpword  $L56
632         .gpword  $L56
633         .gpword  $L56
634         .gpword  $L56
635         .gpword  $L56
636         .gpword  $L56
637         .gpword  $L56
638         .gpword  $L56
639         .gpword  $L56
640         .gpword  $L56
641         .gpword  $L56
642         .gpword  $L56
643         .gpword  $L56
644         .gpword  $L56
645         .gpword  $L56
646         .gpword  $L56
647         .gpword  $L56
648         .gpword  $L51
649         .gpword  $L52
650         .gpword  $L56
651         .gpword  $L56
652         .gpword  $L56
653         .gpword  $L56
654         .gpword  $L56
655         .gpword  $L53
656         .text
657 $L50:
658         la       $t9,executeVersion
659         jal      $ra,$t9

```

```

660         sw      $v0,176($fp)
661         li      $v0,1                      # 0x1
662         sw      $v0,172($fp)
663         b       $L45
664 $L51:
665         la      $t9,executeHelp
666         jal     $ra,$t9
667         sw      $v0,176($fp)
668         li      $v0,1                      # 0x1
669         sw      $v0,172($fp)
670         b       $L45
671 $L52:
672         lw      $v0,optarg
673         sw      $v0,32($fp)
674         b       $L45
675 $L53:
676         lw      $v0,optarg
677         sw      $v0,36($fp)
678         b       $L45
679 $L54:
680         lw      $v0,optarg
681         sw      $v0,40($fp)
682         b       $L45
683 $L55:
684         lw      $v0,optarg
685         sw      $v0,44($fp)
686         b       $L45
687 $L56:
688         li      $v0,1                      # 0x1
689         sw      $v0,168($fp)
690         b       $L45
691 $L46:
692         lw      $v1,168($fp)
693         li      $v0,1                      # 0x1
694         bne     $v1,$v0,$L58
695         la      $a0,___sF+176
696         la      $a1,$LC29
697         la      $t9,fprintf
698         jal     $ra,$t9
699         li      $v0,2                      # 0x2
700         sw      $v0,196($fp)
701         b       $L43
702 $L58:
703         lw      $v1,172($fp)
704         li      $v0,1                      # 0x1
705         bne     $v1,$v0,$L59
706         lw      $v0,176($fp)
707         sw      $v0,196($fp)
708         b       $L43
709 $L59:
710         lw      $v0,40($fp)
711         beq     $v0,$zero,$L60
712         addu    $v0,$fp,188
713         lw      $a0,40($fp)

```

```

714         move    $a1,$v0
715         li      $a2,10                      # 0xa
716         la      $t9,strtoul
717         jal     $ra,$t9
718         sw      $v0,ibytes
719         lw      $v0,ibytes
720         bne     $v0,$zero,$L60
721         la      $a0,___sF+176
722         la      $a1,$LC30
723         la      $t9,fprintf
724         jal     $ra,$t9
725         li      $v1,6                      # 0x6
726         sw      $v1,196($fp)
727         b       $L43
728 $L60:
729         lw      $v0,44($fp)
730         beq     $v0,$zero,$L62
731         addu    $v0,$fp,192
732         lw      $a0,44($fp)
733         move    $a1,$v0
734         li      $a2,10                      # 0xa
735         la      $t9,strtoul
736         jal     $ra,$t9
737         sw      $v0,obytes
738         lw      $v0,obytes
739         bne     $v0,$zero,$L62
740         la      $a0,___sF+176
741         la      $a1,$LC31
742         la      $t9,fprintf
743         jal     $ra,$t9
744         li      $v0,6                      # 0x6
745         sw      $v0,196($fp)
746         b       $L43
747 $L62:
748         lw      $v0,32($fp)
749         bne     $v0,$zero,$L64
750         lw      $v0,36($fp)
751         bne     $v0,$zero,$L64
752         move    $a0,$zero
753         li      $a1,1                      # 0x1
754         li      $a2,1                      # 0x1
755         la      $t9,executeWithDefaultParameter
756         jal     $ra,$t9
757         sw      $v0,196($fp)
758         b       $L43
759 $L64:
760         lw      $v0,32($fp)
761         beq     $v0,$zero,$L65
762         lw      $v0,36($fp)
763         bne     $v0,$zero,$L65
764         la      $a0,$LC32
765         lw      $a1,32($fp)
766         la      $t9,strcmp
767         jal     $ra,$t9

```

```

768         bne     $v0,$zero,$L66
769         move    $a0,$zero
770         li      $a1,1                      # 0x1
771         li      $a2,1                      # 0x1
772         la      $t9,executeWithDefaultParameter
773         jal     $ra,$t9
774         sw      $v0,196($fp)
775         b       $L43
776 $L66:
777         lw      $a0,32($fp)
778         move    $a1,$zero
779         li      $a2,1                      # 0x1
780         la      $t9,executeWithDefaultParameter
781         jal     $ra,$t9
782         sw      $v0,196($fp)
783         b       $L43
784 $L65:
785         lw      $v0,32($fp)
786         bne     $v0,$zero,$L68
787         lw      $v0,36($fp)
788         beq     $v0,$zero,$L68
789         la      $a0,$LC32
790         lw      $a1,36($fp)
791         la      $t9,strcmp
792         jal     $ra,$t9
793         bne     $v0,$zero,$L69
794         move    $a0,$zero
795         li      $a1,1                      # 0x1
796         li      $a2,1                      # 0x1
797         la      $t9,executeWithDefaultParameter
798         jal     $ra,$t9
799         sw      $v0,196($fp)
800         b       $L43
801 $L69:
802         lw      $a0,36($fp)
803         li      $a1,1                      # 0x1
804         move    $a2,$zero
805         la      $t9,executeWithDefaultParameter
806         jal     $ra,$t9
807         sw      $v0,196($fp)
808         b       $L43
809 $L68:
810         lw      $v0,32($fp)
811         beq     $v0,$zero,$L71
812         lw      $v0,36($fp)
813         beq     $v0,$zero,$L71
814         la      $a0,$LC32
815         lw      $a1,32($fp)
816         la      $t9,strcmp
817         jal     $ra,$t9
818         bne     $v0,$zero,$L72
819         la      $a0,$LC32
820         lw      $a1,36($fp)
821         la      $t9,strcmp

```



```

822     jal    $ra,$t9
823     bne    $v0,$zero,$L72
824     move   $a0,$zero
825     li     $a1,1                # 0x1
826     li     $a2,1                # 0x1
827     la     $t9,executeWithDefaultParameter
828     jal    $ra,$t9
829     sw     $v0,196($fp)
830     b      $L43
831 $L72:
832     la     $a0,$LC32
833     lw     $a1,32($fp)
834     la     $t9,strcmp
835     jal    $ra,$t9
836     bne    $v0,$zero,$L73
837     la     $a0,$LC32
838     lw     $a1,36($fp)
839     la     $t9,strcmp
840     jal    $ra,$t9
841     beq    $v0,$zero,$L73
842     lw     $a0,36($fp)
843     li     $a1,1                # 0x1
844     move   $a2,$zero
845     la     $t9,executeWithDefaultParameter
846     jal    $ra,$t9
847     sw     $v0,196($fp)
848     b      $L43
849 $L73:
850     la     $a0,$LC32
851     lw     $a1,32($fp)
852     la     $t9,strcmp
853     jal    $ra,$t9
854     beq    $v0,$zero,$L74
855     la     $a0,$LC32
856     lw     $a1,36($fp)
857     la     $t9,strcmp
858     jal    $ra,$t9
859     bne    $v0,$zero,$L74
860     lw     $a0,32($fp)
861     move   $a1,$zero
862     li     $a2,1                # 0x1
863     la     $t9,executeWithDefaultParameter
864     jal    $ra,$t9
865     sw     $v0,196($fp)
866     b      $L43
867 $L74:
868     lw     $a0,32($fp)
869     lw     $a1,36($fp)
870     la     $t9,executeWithParameters
871     jal    $ra,$t9
872     sw     $v0,196($fp)
873     b      $L43
874 $L71:
875     la     $a0,___sF+176

```

```

876      la      $a1,$LC29
877      la      $t9,fprintf
878      jal     $ra,$t9
879      li      $v1,2                      # 0x2
880      sw      $v1,196($fp)
881 $L43:
882      lw      $v0,196($fp)
883      move    $sp,$fp
884      lw      $ra,216($sp)
885      lw      $fp,212($sp)
886      addu    $sp,$sp,224
887      j       $ra
888      .end    executeByMenu
889      .size   executeByMenu, .-executeByMenu
890      .rdata
891      .align  2
892 $LC33:
893      .ascii  "[Error] Cantidad m\303\241xima de par\303\241
          metros inco"
894      .ascii  "rrecta: %d \n\000"
895      .text
896      .align  2
897      .globl  main
898      .ent    main
899 main:
900      .frame  $fp,48,$ra                  # vars= 8, regs= 3/0,
          args= 16, extra= 8
901      .mask   0xd0000000,-8
902      .fmask  0x00000000,0
903      .set    noreorder
904      .cpld   $t9
905      .set    reorder
906      subu    $sp,$sp,48
907      .cpstore 16
908      sw      $ra,40($sp)
909      sw      $fp,36($sp)
910      sw      $gp,32($sp)
911      move    $fp,$sp
912      sw      $a0,48($fp)
913      sw      $a1,52($fp)
914      lw      $v0,48($fp)
915      slt     $v0,$v0,10
916      bne     $v0,$zero,$L76
917      la      $a0,___sF+176
918      la      $a1,$LC33
919      lw      $a2,48($fp)
920      la      $t9,fprintf
921      jal     $ra,$t9
922      li      $v0,1                      # 0x1
923      sw      $v0,24($fp)
924      b       $L75
925 $L76:
926      lw      $a0,48($fp)
927      lw      $a1,52($fp)

```

```

928      la      $t9,executeByMenu
929      jal     $ra,$t9
930      sw      $v0,24($fp)
931 $L75:
932      lw      $v0,24($fp)
933      move    $sp,$fp
934      lw      $ra,40($sp)
935      lw      $fp,36($sp)
936      addu    $sp,$sp,48
937      j       $ra
938      .end    main
939      .size   main,.-main
940      .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520) "

```

3.3. Código MIPS32 función palindrome()

1 ACA CÓDIGO MIPS PALINDROME

3.4. Stack frame de funciones

Función: toLowerCase()

| mem.pos | register name |
|---------|---------------|
| 24 | - |
| 20 | fp |
| 16 | gp |
| 12 | - |
| 8 | a0 |
| 4 | - |
| 0 | - |

Cuadro 1: Stack frame.

Función: verifyPalindromic()

| mem.pos | register name |
|---------|----------------------------|
| 76 | a1=quantityCharacterInWord |
| 72 | a0 |
| 64 | ra |
| 60 | fp |
| 56 | gp |
| 52 | FALSE |

Cuadro 2: Stack frame.

toLowerCase -> ifNotLower

| mem.pos | register name |
|---------|----------------------------|
| 76 | a1=quantityCharacterInWord |
| 72 | a0 |
| 64 | ra |
| 60 | fp |
| 56 | gp |
| 52 | TRUE |

Cuadro 3: Stack frame.TRUE: 1.

| mem.pos | register name |
|---------|---------------|
| 52 | TRUE |

Cuadro 4: Stack frame.VerifyWhenOneCharacter: quantityCharacterInWord == 1.

| mem.pos | register name |
|---------|---------------|
| 52 | TRUE |

Cuadro 5: Stack frame.VerifyWhenOneCharacter: quantityCharacterInWord != 1.

| mem.pos | register name |
|---------|---------------|
| 24 | - |
| 20 | fp |
| 16 | gp |
| 12 | - |
| 8 | a0=character |
| 4 | - |
| 0 | - |

Cuadro 6: Stack frame.VerifyWhenTwoCharacteres: quantityCharacterInWord == 2.

| mem.pos | register name |
|---------|-------------------|
| 8 | a0=character + 32 |

Cuadro 7: Stack frame.VerifyWhenTwoCharacteres: character is >=65 <=90.

4. Ejecución

A continuación algunos de los comandos válidos para la ejecución del programa:
Comandos usando un archivo de entrada y otro de salida

```
$ tpl -i input.txt -o output.txt
```

```
$ tpl --input input.txt --output output.txt
```

Comando para la salida standard

```
$ tpl -i input.txt
```

Comando para el ingreso standard

```
$ tpl -o output.txt
```

Por defecto los tamaños del buffer in y buffer out son 1 byte. puede especificar el tamaño a usar los mismos en la llamada.

```
$ tpl -i input.txt -o output.txt -I 10 -O 10
```

-I: indica el tamaño (bytes) a usar por el buffer in

-O: indica el tamaño (bytes) a usar por el buffer out

4.1. Comandos para ejecución

Desde el netBSD ejecutar:
Para compilar el código

```
$ gcc -Wall -o tpl tpl.c mips.S
```

-Wall: activa los mensajes de warning

-o: indica el archivo de salida.

Para obtener el código MIPS32

```
$ gcc -Wall -O0 -S -mnames tpl.c
```

-S: detiene el compilador luego de generar el código assembly

-mnames: indica al compilador que genere la salida con nombre de registros

-O0: indica al compilador que no aplique optimizaciones.

4.2. Análisis sobre tiempo de ejecución

Comando para la medición del tiempo (time):

```
$ time ./tpl -i ../input-large.txt -I 10 -O 10
```

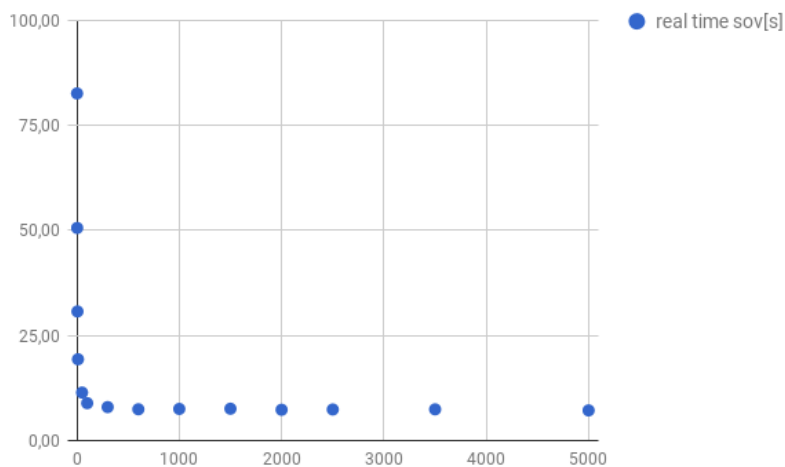


Figura 1: Gráfico de incidencia del buffer

Se midieron y se tuvieron en cuenta los tiempo transcurridos entre distintas ejecuciones cambiando los parámetros de entrada de buffer in y buffer out. Para medir se usó la instrucción "time" la cual arroja los tiempos efectivamente consumidos por el CPU en la ejecución del programa. A continuación una tabla con los valores medidos:

Tamaño de archivo usado aproximadamente 1,70 MB.

Tamaño de línea en archivo aproximadamente: 1 byte * 450 char = 450 byte(caracteres/línea).

| id | stream input | stream output | real time[s] | user time[s] | sys time[s] |
|----|--------------|---------------|--------------|--------------|-------------|
| 1 | 1 | 1 | 82,53 | 9,74 | 36,68 |
| 2 | 2 | 2 | 50,55 | 8,16 | 21,11 |
| 3 | 3 | 2 | 30,70 | 7,78 | 8,72 |
| 4 | 5 | 5 | 19,34 | 6,94 | 4,67 |
| 5 | 10 | 10 | 11,42 | 7,10 | 1,10 |
| 6 | 50 | 50 | 8,90 | 6,85 | 0,66 |
| 7 | 100 | 100 | 7,97 | 6,97 | 0,34 |
| 8 | 300 | 300 | 7,45 | 6,76 | 0,22 |
| 9 | 600 | 600 | 7,53 | 6,90 | 0,18 |
| 10 | 1000 | 1000 | 7,58 | 6,89 | 0,25 |
| 11 | 1500 | 1500 | 7,33 | 6,84 | 0,24 |
| 12 | 2000 | 2000 | 7,40 | 6,86 | 0,23 |
| 13 | 2500 | 2500 | 7,43 | 6,94 | 0,11 |
| 14 | 3500 | 3500 | 7,18 | 6,70 | 0,16 |

Cuadro 8: Valores de exe medidos(time).

Cómo puede verse en la figura las ejecuciones iniciales con valores bajos de lectura

y escritura(buffer 1 byte) tienen tiempos de respuesta del programa elevados; mientras que a medida que se aumenta el tamaño del buffer los tiempos van creciendo hasta un límite asintótico alrededor de 7 segundos.

Es de notar que un pequeño aumento en el tamaño del buffer(in/out) aumenta la performance considerablemente.

4.3. Comandos para ejecución de tests

Comando para ejecutar el test automático

```
$ bash test-automatic.sh
```

La salida debería ser la siguiente(todos los test OK):

```
ACA CÓDIGO DEL BASH
```

5. Conclusiones

A través del presente trabajo se logro realizar una implementación pequeña de un programa c y assembly MIPS32. La invocación desde un programa assembly a un programa c; la implementación de una función malloc, free y realloc en código assembly, sin hacer uso de la implementación c. La forma de llamar a funciones de

Por otro lado se logró familiarizarse con la implementación de assembly MIPS y con la ABI.

La implementación de la función palindroma con un buffer permitió ver que en función de la cantidad de caracteres leídos cada vez, el tiempo de ejecución del programa disminuía considerablemente. Al mismo tiempo la mejora en el tiempo de ejecución tiene un límite a partir del cual un aumento en el tamaño del buffer no garantiza ganancia en la ejecución del programa.

Referencias

- [1] Intel Technology & Research, “Hyper-Threading Technology,” 2006, <http://www.intel.com/technology/hyperthread/>.
- [2] J. L. Hennessy and D. A. Patterson, “Computer Architecture. A Quantitative Approach,” 3ra Edición, Morgan Kaufmann Publishers, 2000.
- [3] J. Larus and T. Ball, “Rewriting Executable Files to Measure Program Behavior,” Tech. Report 1083, Univ. of Wisconsin, 1992.
<https://es.wikipedia.org/wiki/Pal>