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 palindroma. La función estará escrita en assembly MIPS.

Los streams serán leídos y escritos de a bloques de memoría 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

```
/*
 ============================================================================

 Name        : tp1.c
 Author      : Grupo orga 66.20
 Version     : 1
 Copyright   : Orga6620 - Tp1
 Description : Trabajo practico 1: Programacion MIPS
 ============================================================================

 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include <unistd.h>
#include "process.h"

#define VERSION "1.0"

#define FALSE 0
#define TRUE 1

size_t ibytes = 1;
size_t obytes = 1;

enum ParameterState {
         OKEY = 0, INCORRECT_QUANTITY_PARAMS = 1,
             INCORRECT_MENU = 2, ERROR_FILE = 3, ERROR_MEMORY =
             4, ERROR_READ = 5, ERROR_BYTES = 6
};
```

```
29
30 int executeHelp() {
31         fprintf(stdout, "Usage: \n");
32         fprintf(stdout, "       tp1 -h \n");
33         fprintf(stdout, "       tp1 -V \n");
34         fprintf(stdout, "       tp1 [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, "       tp1 -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
       , int isOutputDefault) {
55         FILE * fileInput = NULL;
56         FILE * fileOutput = NULL;
57
58         if (isInputDefault == TRUE && isOutputDefault == TRUE)
               {
59                 fileInput = stdin;
60                 fileOutput = stdout;
61         } else {
62                 if (isInputDefault == TRUE) {
63                         fileInput = stdin;
64
65                         fileOutput = fopen(path, "w"); // Opens
                             a text file for writing. Pace the
                            content.
66                         if (fileOutput == NULL) {
67                                 fprintf(stderr, "[Error] El
                                     archivo de output no pudo
                                     ser abierto para escritura:
                                     %s \n", path);
68                                 return ERROR_FILE;
```

```
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          if (isInputDefault == FALSE || isOutputDefault == FALSE
                ) {
87                  if (isInputDefault == TRUE) {
88                          if (fileOutput != NULL) {
89                                  int result = fclose(fileOutput)
                                       ;
90                                  if (result == EOF) {
91                                          fprintf(stderr, "[
                                               Warning] El archivo
                                               de output no pudo
                                               ser cerrado
                                               correctamente: %s \n
                                               ", path);
92                                          return ERROR_FILE;
93                                  }
94                          }
95                  } else {
96                          if (fileInput != NULL) {
97                                  int result = fclose(fileInput);
98                                  if (result == EOF) {
99                                          fprintf(stderr, "[
                                               Warning] El archivo
                                               de input no pudo ser
                                                cerrado
                                               correctamente: %s \n
                                               ", path);
100                                         return ERROR_FILE;
101                                 }
102                         }
103                 }
104         }
```

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

```
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,
                        TRUE);
161         }
162
163         char * inputValue = NULL;
164         char * outputValue = NULL;
165         char * iBufBytes = NULL;
166         char * oBufBytes = NULL;
167
168         /* Una cadena que lista las opciones cortas validas */
169         const char* const smallOptions = "Vhi:o:I:O:";
170
171         /* Una estructura de varios arrays describiendo los
                valores largos */
172         const struct option longOptions[] = {
173                 {"version",             no_argument,
                                0,  'V' },
174                 {"help",                no_argument,
                                        0,  'h' },
175                 {"input",               required_argument,  0,
                        'i' }, // optional_argument
176                 {"output",              required_argument,  0,
                        'o' },
177                 {"ibuf-bytes",  required_argument,      0,  'I'
                        },
178                 {"obuf-bytes",  required_argument,      0,  'O'
                        },
179                 {0,                     0,
                                                 0,   0  }
180         };
181
182         int incorrectOption = FALSE;
183         int finish = FALSE;
184         int result = OKEY;
185         int longIndex = 0;
186         char opt = 0;
187
188         while ((opt = getopt_long(argc, argv, smallOptions,
```

```
                                    longOptions, &longIndex ))
                                        != -1 && incorrectOption
                                        == FALSE && finish ==
                                        FALSE) {
                switch (opt) {
                        case 'V' :
                                result = executeVersion();
                                finish = TRUE;
                                break;
                        case 'h' :
                                result = executeHelp();
                                finish = TRUE;
                                break;
                        case 'i' :
                                inputValue = optarg;
                                break;
                        case 'o' :
                                outputValue = optarg;
                                break;
                        case 'I' :
                                iBufBytes = optarg;
                                break;
                        case 'O' :
                                oBufBytes = optarg;
                                break;
                        default:
                                incorrectOption = TRUE;
                }
        }

        if (incorrectOption == TRUE) {
                fprintf(stderr, "[Error] Incorrecta option de
                    menu.\n");
                return INCORRECT_MENU;
        }

        if (finish == TRUE) {
                return result;
        }

        if (iBufBytes != NULL) {
                char *finalPtr;
                ibytes = strtoul(iBufBytes, &finalPtr, 10);
                if (ibytes == 0) {
                        fprintf(stderr, "[Error] Incorrecta
                            cantidad de bytes para el buffer de
                            entrada.\n");
                        return ERROR_BYTES;
                }
        }

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

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:
```

```
        .ascii  "\ttp1 [options] \n\000"
        .align  2
$LC4:
        .ascii  "Options: \n\000"
        .align  2
$LC5:
        .ascii  "\t-V, --version\t\tPrint version and quit. \n
            \000"
        .align  2
$LC6:
        .ascii  "\t-h, --help\t    \tPrint this information. \n
            \000"
        .align  2
$LC7:
        .ascii  "\t-i, --input\t    \tLocation of the input
            file. \n\000"
        .align  2
$LC8:
        .ascii  "\t-o, --output\t\tLocation of the output file.
             \n\000"
        .align  2
$LC9:
        .ascii  "\t-I, --ibuf-bytes\tByte-count of the input
            buffer. \n\000"
        .align  2
$LC10:
        .ascii  "\t-O, --obuf-bytes\tByte-count of the output
            buffer. \n\000"
        .align  2
$LC11:
        .ascii  "Examples: \n\000"
        .align  2
$LC12:
        .ascii  "\ttp1 -i ~/input -o ~/output \n\000"
        .text
        .align  2
        .globl  executeHelp
        .ent    executeHelp
executeHelp:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0,
            args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        la      $a0,__sF+88
        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,
                args= 16, extra= 8
148         .mask   0xd0000000,-8
149         .fmask  0x00000000,0
150         .set    noreorder
151         .cpload $t9
152         .set    reorder
153         subu    $sp,$sp,40
154         .cprestore 16
155         sw      $ra,32($sp)
156         sw      $fp,28($sp)
157         sw      $gp,24($sp)
158         move    $fp,$sp
159         la      $a0,__sF+88
160         la      $a1,$LC13
161         la      $a2,$LC14
162         la      $t9,fprintf
163         jal     $ra,$t9
164         move    $v0,$zero
165         move    $sp,$fp
166         lw      $ra,32($sp)
167         lw      $fp,28($sp)
168         addu    $sp,$sp,40
169         j       $ra
170         .end    executeVersion
171         .size   executeVersion, .-executeVersion
172         .rdata
173         .align  2
174 $LC15:
175         .ascii  "w\000"
176         .align  2
177 $LC16:
178         .ascii  "[Error] El archivo de output no pudo ser
                abierto para es"
179         .ascii  "critura: %s \n\000"
180         .align  2
181 $LC17:
182         .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         .cprestore 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
```

```
        sw      $v0,24($fp)
        lw      $a0,72($fp)
        la      $a1,$LC15
        la      $t9,fopen
        jal     $ra,$t9
        sw      $v0,28($fp)
        lw      $v0,28($fp)
        bne     $v0,$zero,$L21
        la      $a0,__sF+176
        la      $a1,$LC16
        lw      $a2,72($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        li      $v0,3                   # 0x3
        sw      $v0,48($fp)
        b       $L19
$L22:
        lw      $a0,72($fp)
        la      $a1,$LC17
        la      $t9,fopen
        jal     $ra,$t9
        sw      $v0,24($fp)
        lw      $v0,24($fp)
        bne     $v0,$zero,$L25
        la      $a0,__sF+176
        la      $a1,$LC18
        lw      $a2,72($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        li      $v0,3                   # 0x3
        sw      $v0,48($fp)
        b       $L19
$L25:
        la      $v0,__sF+88
        sw      $v0,28($fp)
$L21:
        lw      $v0,24($fp)
        lh      $v0,14($v0)
        sw      $v0,32($fp)
        lw      $v0,28($fp)
        lh      $v0,14($v0)
        sw      $v0,36($fp)
        lw      $a0,32($fp)
        lw      $a1,ibytes
        lw      $a2,36($fp)
        lw      $a3,obytes
        la      $t9,palindrome
        jal     $ra,$t9
        sw      $v0,40($fp)
        lw      $v0,76($fp)
        beq     $v0,$zero,$L27
        lw      $v0,80($fp)
        bne     $v0,$zero,$L26
$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, .-
                executeWithDefaultParameter
337         .align  2
338         .globl  executeWithParameters
339         .ent    executeWithParameters
```

```
executeWithParameters:
        .frame  $fp,72,$ra              # vars= 32, regs= 3/0,
           args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,72
        .cprestore 16
        sw      $ra,64($sp)
        sw      $fp,60($sp)
        sw      $gp,56($sp)
        move    $fp,$sp
        sw      $a0,72($fp)
        sw      $a1,76($fp)
        lw      $a0,72($fp)
        la      $a1,$LC17
        la      $t9,fopen
        jal     $ra,$t9
        sw      $v0,24($fp)
        lw      $v0,24($fp)
        bne     $v0,$zero,$L35
        la      $a0,__sF+176
        la      $a1,$LC18
        lw      $a2,72($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        li      $v0,3                   # 0x3
        sw      $v0,52($fp)
        b       $L34
$L35:
        lw      $a0,76($fp)
        la      $a1,$LC15
        la      $t9,fopen
        jal     $ra,$t9
        sw      $v0,28($fp)
        lw      $v0,28($fp)
        bne     $v0,$zero,$L36
        la      $a0,__sF+176
        la      $a1,$LC16
        lw      $a2,76($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        lw      $a0,24($fp)
        la      $t9,fclose
        jal     $ra,$t9
        sw      $v0,32($fp)
        lw      $v1,32($fp)
        li      $v0,-1                  # 0xffffffffffffffff
        bne     $v1,$v0,$L37
        la      $a0,__sF+176
        la      $a1,$LC20
        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:
```

```
        lw      $v0,44($fp)
        beq     $v0,$zero,$L42
        li      $v0,3                   # 0x3
        sw      $v0,52($fp)
        b       $L34
$L42:
        lw      $v0,40($fp)
        sw      $v0,52($fp)
$L34:
        lw      $v0,52($fp)
        move    $sp,$fp
        lw      $ra,64($sp)
        lw      $fp,60($sp)
        addu    $sp,$sp,72
        j       $ra
        .end    executeWithParameters
        .size   executeWithParameters, .-executeWithParameters
        .rdata
        .align  2
$LC22:
        .ascii  "version\000"
        .align  2
$LC23:
        .ascii  "help\000"
        .align  2
$LC24:
        .ascii  "input\000"
        .align  2
$LC25:
        .ascii  "output\000"
        .align  2
$LC26:
        .ascii  "ibuf-bytes\000"
        .align  2
$LC27:
        .ascii  "obuf-bytes\000"
        .data
        .align  2
$LC28:
        .word   $LC22
        .word   0
        .word   0
        .word   86
        .word   $LC23
        .word   0
        .word   0
        .word   104
        .word   $LC24
        .word   1
        .word   0
        .word   105
        .word   $LC25
        .word   1
        .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
```

```
        beq     $v0,$zero,$L56
        lw      $v0,200($fp)
        sll     $v1,$v0,2
        la      $v0,$L57
        addu    $v0,$v1,$v0
        lw      $v0,0($v0)
        .cpadd  $v0
        j       $v0
        .rdata
        .align  2
$L57:
        .gpword $L54
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L55
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L50
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L51
        .gpword $L52
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L56
        .gpword $L53
        .text
$L50:
        la      $t9,executeVersion
        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)
```

```
        move    $a1,$v0
        li      $a2,10                  # 0xa
        la      $t9,strtoul
        jal     $ra,$t9
        sw      $v0,ibytes
        lw      $v0,ibytes
        bne     $v0,$zero,$L60
        la      $a0,__sF+176
        la      $a1,$LC30
        la      $t9,fprintf
        jal     $ra,$t9
        li      $v1,6                   # 0x6
        sw      $v1,196($fp)
        b       $L43
$L60:
        lw      $v0,44($fp)
        beq     $v0,$zero,$L62
        addu    $v0,$fp,192
        lw      $a0,44($fp)
        move    $a1,$v0
        li      $a2,10                  # 0xa
        la      $t9,strtoul
        jal     $ra,$t9
        sw      $v0,obytes
        lw      $v0,obytes
        bne     $v0,$zero,$L62
        la      $a0,__sF+176
        la      $a1,$LC31
        la      $t9,fprintf
        jal     $ra,$t9
        li      $v0,6                   # 0x6
        sw      $v0,196($fp)
        b       $L43
$L62:
        lw      $v0,32($fp)
        bne     $v0,$zero,$L64
        lw      $v0,36($fp)
        bne     $v0,$zero,$L64
        move    $a0,$zero
        li      $a1,1                   # 0x1
        li      $a2,1                   # 0x1
        la      $t9,executeWithDefaultParameter
        jal     $ra,$t9
        sw      $v0,196($fp)
        b       $L43
$L64:
        lw      $v0,32($fp)
        beq     $v0,$zero,$L65
        lw      $v0,36($fp)
        bne     $v0,$zero,$L65
        la      $a0,$LC32
        lw      $a1,32($fp)
        la      $t9,strcmp
        jal     $ra,$t9
```

```
        bne     $v0,$zero,$L66
        move    $a0,$zero
        li      $a1,1                   # 0x1
        li      $a2,1                   # 0x1
        la      $t9,executeWithDefaultParameter
        jal     $ra,$t9
        sw      $v0,196($fp)
        b       $L43
$L66:
        lw      $a0,32($fp)
        move    $a1,$zero
        li      $a2,1                   # 0x1
        la      $t9,executeWithDefaultParameter
        jal     $ra,$t9
        sw      $v0,196($fp)
        b       $L43
$L65:
        lw      $v0,32($fp)
        bne     $v0,$zero,$L68
        lw      $v0,36($fp)
        beq     $v0,$zero,$L68
        la      $a0,$LC32
        lw      $a1,36($fp)
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L69
        move    $a0,$zero
        li      $a1,1                   # 0x1
        li      $a2,1                   # 0x1
        la      $t9,executeWithDefaultParameter
        jal     $ra,$t9
        sw      $v0,196($fp)
        b       $L43
$L69:
        lw      $a0,36($fp)
        li      $a1,1                   # 0x1
        move    $a2,$zero
        la      $t9,executeWithDefaultParameter
        jal     $ra,$t9
        sw      $v0,196($fp)
        b       $L43
$L68:
        lw      $v0,32($fp)
        beq     $v0,$zero,$L71
        lw      $v0,36($fp)
        beq     $v0,$zero,$L71
        la      $a0,$LC32
        lw      $a1,32($fp)
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L72
        la      $a0,$LC32
        lw      $a1,36($fp)
        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         .cpload $t9
905         .set    reorder
906         subu    $sp,$sp,48
907         .cprestore 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: toLoweCase()

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

```
$ tp1 -i input.txt -o output.txt
```

```
$ tp1 --input input.txt --output output.txt
```

Comando para la salida standard

```
$ tp1 -i input.txt
```

Comando para el ingreso standard

```
$ tp1 -o output.txt
```

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

```
$ tp1 -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 tp1 tp1.c mips.S
```

-Wall: activa los mensajes de warnning
-o: indica el archivo de salida.

Para obtener el código MIPS32

```
 $ gcc -Wall -O0 -S -mrnames tp1.c
```

-S: detiene el compilador luego de generar el código assembly
-mrnames: 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 ./tp1 -i ../input-large.txt -I 10 -O 10
```

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 apróximadamente 337 kB.
Tamaño de línea en archivo apróximadamente: 1 byte * 450 char = 450 byte(caracteres/línea).

id	stream input	stream output	real time[ms]	user time[ms]	sys time[ms]
1	1	1	132	52	76
2	2	2	70	36	32
3	5	5	52	36	12
4	10	10	33	20	12
5	50	50	25	20	4
6	100	100	24	20	0
7	300	300	23	20	84
8	600	600	22	28	76
9	1000	1000	22	20	0
10	1500	1500	23	20	0
11	2000	2000	24	24	0
12	2500	2500	22	24	0
13	3500	3500	21	12	8
14	5000	5000	23	20	0

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 limite asintótico alrededor de 22 ms.
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
```

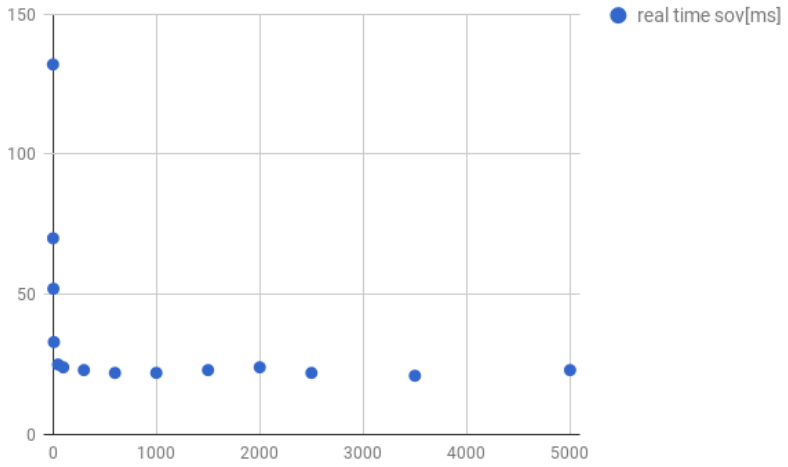

Figura 1: Gráfico de incidencia del buffer

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 disminuia 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 Mesure Program Behavior,” Tech. Report 1083, Univ. of Wisconsin, 1992.

https://es.wikipedia.org/wiki/Pal