

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.

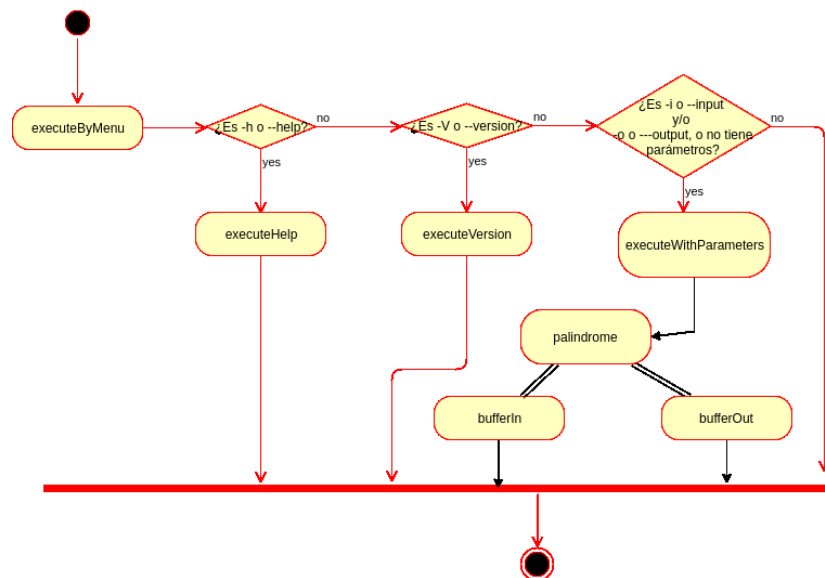


Figura 1: Diagrama de actividad

- 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
  
```

```

8  =====
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  };
33
34  int executeHelp() {
35      fprintf(stdout, "Usage: \n");
36      fprintf(stdout, "    tpl -h \n");
37      fprintf(stdout, "    tpl -V \n");
38      fprintf(stdout, "    tpl [options] \n");
39      fprintf(stdout, "Options: \n");
40      fprintf(stdout, "    -V, --version          Print
41      version and quit. \n");
42      fprintf(stdout, "    -h, --help            Print
43      this information. \n");
44      fprintf(stdout, "    -i, --input           Location of the input file. \n");
45      fprintf(stdout, "    -o, --output          Location of the output file. \n");
46      fprintf(stdout, "    -I, --ibuf-bytes      Byte-
47      count of the input buffer. \n");
48      fprintf(stdout, "    -O, --obuf-bytes      Byte-
49      count of the output buffer. \n");
50      fprintf(stdout, "Examples: \n");
51      fprintf(stdout, "    tpl -i ~/input -o ~/output \n");
52      ;
53
54      return OKEY;
55  }
56
57  int executeVersion() {
58      fprintf(stdout, "Version: \"%s\" \n", VERSION);
59
60      return OKEY;
61  }

```

```

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     {
60         fileInput = stdin;
61         fileOutput = stdout;
62     } else {
63         if (isInputDefault == TRUE) {
64             fileInput = stdin;
65
66             fileOutput = fopen(path, "w"); // Opens
67                 a text file for writing. Place the
68                 content.
69             if (fileOutput == NULL) {
70                 fprintf(stderr, "[Error] El
71                     archivo de output no pudo
72                     ser abierto para escritura:
73                     %s \n", path);
74                 return ERROR_FILE;
75             }
76         } else {
77             fileInput = fopen(path, "r"); // Opens
78                 an existing text file for reading
79                 purpose.
80             if (fileInput == NULL) {
81                 fprintf(stderr, "[Error] El
82                     archivo de input no pudo ser
83                     abierto para lectura: %s \n
84                     ", path);
85                 return ERROR_FILE;
86             }
87
88             fileOutput = stdout;
89         }
90     }
91
92     int ifd = fileno(fileInput);
93     int ofd = fileno(fileOutput);
94
95     int executeResult = palindrome(ifd, 1, ofd, 1);
96
97     if (isInputDefault == FALSE || isOutputDefault == FALSE)
98     {
99         if (isInputDefault == TRUE) {
100             if (fileOutput != NULL) {
101                 int result = fclose(fileOutput);
102                 ;
103                 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 {
111     FILE * fileInput = fopen(pathInput, "r"); // Opens an
           existing text file for reading purpose.
112     if (fileInput == NULL) {
113         fprintf(stderr, "[Error] El archivo de input no
           pudo ser abierto para lectura: %s \n",
           pathInput);
114         return ERROR_FILE;
115     }
116
117     FILE * fileOutput = fopen(pathOutput, "w"); // Opens a
           text file for writing. Place the content.
118     if (fileOutput == NULL) {
119         fprintf(stderr, "[Error] El archivo de output
           no pudo ser abierto para escritura: %s \n",
           pathOutput);
120
121         int result = fclose(fileInput);
122         if (result == EOF) {
123             fprintf(stderr, "[Warning] El archivo
               de input no pudo ser cerrado
               correctamente: %s \n", pathInput);
124         }
125
126         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
134     int resultFileInputClose = 0; // EOF = -1
135     if (fileInput != NULL) {
136         resultFileInputClose = fclose(fileInput);
137         if (resultFileInputClose == EOF) {
138             fprintf(stderr, "[Warning] El archivo
139                 de input no pudo ser cerrado
140                 correctamente: %s \n", pathInput);
141         }
142     }
143
144     if (fileOutput != NULL) {
145         int result = fclose(fileOutput);
146         if (result == EOF) {
147             fprintf(stderr, "[Warning] El archivo
148                 de output no pudo ser cerrado
149                 correctamente: %s \n", pathOutput);
150             return ERROR_FILE;
151         }
152     }
153
154     if (resultFileInputClose) {
155         return ERROR_FILE;
156     }
157
158     return executeResult;
159 }
160
161 int executeByMenu(int argc, char **argv) {
162     // Always begins with /
163     if (argc == 1) {
164         // Run with default parameters
165         return executeWithDefaultParameter(NULL, TRUE,
166             TRUE);
167     }
168
169     char * inputValue = NULL;
170     char * outputValue = NULL;
171     char * iBufBytes = NULL;
172     char * oBufBytes = NULL;
173
174     /* Una cadena que lista las opciones cortas validas */
175     const char* const smallOptions = "Vhi:o:I:O:";
176
177     /* Una estructura de varios arrays describiendo los
178         valores largos */
179     const struct option longOptions[] = {

```

```

173         {"version",          no_argument,
174         0, 'V' },
175         {"help",             no_argument,
176         0, 'h' },
177         {"input",             required_argument, 0,
178         'i' }, // optional_argument
179         {"output",            required_argument, 0,
180         'o' },
181         {"ibuf-bytes",        required_argument, 0, 'I'
182         },
183         {"obuf-bytes",        required_argument, 0, 'O'
184         },
185         {0,                    0,
186         0, 0 }
187     };
188
189     int incorrectOption = FALSE;
190     int finish = FALSE;
191     int result = OKEY;
192     int longIndex = 0;
193     char opt = 0;
194
195     while ((opt = getopt_long(argc, argv, smallOptions,
196                             longOptions, &longIndex))
197            != -1 && incorrectOption
198            == FALSE && finish ==
199            FALSE) {
200
201         switch (opt) {
202             case 'V' :
203                 result = executeVersion();
204                 finish = TRUE;
205                 break;
206             case 'h' :
207                 result = executeHelp();
208                 finish = TRUE;
209                 break;
210             case 'i' :
211                 inputValue = optarg;
212                 break;
213             case 'o' :
214                 outputValue = optarg;
215                 break;
216             case 'I' :
217                 iBufBytes = optarg;
218                 break;
219             case 'O' :
220                 oBufBytes = optarg;
221                 break;
222             default:
223                 incorrectOption = TRUE;
224         }
225     }
226
227     if (incorrectOption == TRUE) {

```

```

217         fprintf(stderr, "[Error] Incorrecta option de
218             menu.\n");
219         return INCORRECT_MENU;
220     }
221     if (finish == TRUE) {
222         return result;
223     }
224
225     if (iBufBytes != NULL) {
226         char *finalPtr;
227         ibytes = strtoul(iBufBytes, &finalPtr, 10);
228         if (ibytes == 0) {
229             fprintf(stderr, "[Error] Incorrecta
230                 cantidad de bytes para el buffer de
231                 entrada.\n");
232             return ERROR_BYTES;
233         }
234     }
235
236     if (oBufBytes != NULL) {
237         char *finalPtr;
238         obytes = strtoul(oBufBytes, &finalPtr, 10);
239         if (obytes == 0) {
240             fprintf(stderr, "[Error] Incorrecta
241                 cantidad de bytes para el buffer de
242                 salida.\n");
243             return ERROR_BYTES;
244         }
245     }
246
247     if (inputValue == NULL && outputValue == NULL) {
248         return executeWithDefaultParameter(NULL, TRUE,
249             TRUE);
250     }
251
252     // / -i fileInput
253     if (inputValue != NULL && outputValue == NULL) {
254         if (strcmp("-",inputValue) == 0) {
255             return executeWithDefaultParameter(NULL
256                 , TRUE, TRUE);
257         } else {
258             return executeWithDefaultParameter(
259                 inputValue, FALSE, TRUE);
260         }
261     }
262
263     // / -o fileOutput
264     if (inputValue == NULL && outputValue != NULL) {
265         if (strcmp("-",outputValue) == 0) {
266             return executeWithDefaultParameter(NULL
267                 , TRUE, TRUE);
268         } else {

```



```

261         return executeWithDefaultParameter(
262             outputValue, TRUE, FALSE);
263     }
264 }
265 if (inputValue != NULL && outputValue != NULL) {
266     if (strcmp("-",inputValue) == 0 && strcmp("-",
267         outputValue) == 0) {
268         return executeWithDefaultParameter(NULL
269             , TRUE, TRUE);
270     }
271     if (strcmp("-",inputValue) == 0 && strcmp("-",
272         outputValue) != 0) {
273         return executeWithDefaultParameter(
274             outputValue, TRUE, FALSE);
275     }
276     if (strcmp("-",inputValue) != 0 && strcmp("-",
277         outputValue) == 0) {
278         return executeWithDefaultParameter(
279             inputValue, FALSE, TRUE);
280     }
281     return executeWithParameters(inputValue,
282         outputValue);
283 }
284
285 fprintf(stderr, "[Error] Incorrecta option de menu.\n")
286 ;
287 return INCORRECT_MENU;
288 }
289
290 int main(int argc, char **argv) {
291     // / -i lalala.txt -o pepe.txt -I 2 -O 3 => 9
292     // parameters as maximum
293     if (argc > 9) {
294         fprintf(stderr, "[Error] Cantidad máxima de
295             parámetros incorrecta: %d \n", argc);
296         return INCORRECT_QUANTITY_PARAMS;
297     }
298     return executeByMenu(argc, argv);
299 }

```

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:

```



```

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     .cpld   $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
179             abierto para es"
180      .ascii  "critura: %s \n\000"
181      .align  2
182 $LC17:
183      .ascii  "r\000"
184      .align  2
185 $LC18:
186      .ascii  "[Error] El archivo de input no pudo ser
187             abierto para lec"
188      .ascii  "tura: %s \n\000"
189      .align  2
190 $LC19:
191      .ascii  "[Warning] El archivo de output no pudo ser
192             cerrado corre"
193      .ascii  "ctamente: %s \n\000"
194      .align  2
195 $LC20:
196      .ascii  "[Warning] El archivo de input no pudo ser
197             cerrado correc"
198      .ascii  "tamente: %s \n\000"
199      .text
200      .align  2
201      .globl  executeWithDefaultParameter
202      .ent    executeWithDefaultParameter
203 executeWithDefaultParameter:
204      .frame  $fp,72,$ra          # vars= 32, regs= 3/0,
205             args= 16, extra= 8
206      .mask   0xd0000000,-8
207      .fmask  0x00000000,0
208      .set    noreorder
209      .cpload $t9
210      .set    reorder
211      subu    $sp,$sp,72
212      .cpstore 16
213      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,abytes
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                    # 0xfffffffffffffffffff
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                    # 0xfffffffffffffffffff
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
341 executeWithParameters:
342         .frame   $fp, 72, $ra                # vars= 32, regs= 3/0,
343                 args= 16, extra= 8
344         .mask    0xd0000000, -8
345         .fmask   0x00000000, 0
346         .set     noreorder
347         .cpload  $t9
348         .set     reorder
349         subu     $sp, $sp, 72
350         .cpstore 16
351         sw      $ra, 64($sp)
352         sw      $fp, 60($sp)
353         sw      $gp, 56($sp)
354         move     $fp, $sp
355         sw      $a0, 72($fp)
356         sw      $a1, 76($fp)
357         lw      $a0, 72($fp)
358         la      $a1, $LC17
359         la      $t9, fopen
360         jal     $ra, $t9
361         sw      $v0, 24($fp)
362         lw      $v0, 24($fp)
363         bne     $v0, $zero, $L35
364         la      $a0, __sF+176
365         la      $a1, $LC18
366         lw      $a2, 72($fp)
367         la      $t9, fprintf
368         jal     $ra, $t9
369         li      $v0, 3                      # 0x3
370         sw      $v0, 52($fp)

```



```

369      b      $L34
370 $L35:
371      lw      $a0,76($fp)
372      la      $a1,$LC15
373      la      $t9,fopen
374      jal     $ra,$t9
375      sw      $v0,28($fp)
376      lw      $v0,28($fp)
377      bne     $v0,$zero,$L36
378      la      $a0,___sF+176
379      la      $a1,$LC16
380      lw      $a2,76($fp)
381      la      $t9,fprintf
382      jal     $ra,$t9
383      lw      $a0,24($fp)
384      la      $t9,fclose
385      jal     $ra,$t9
386      sw      $v0,32($fp)
387      lw      $v1,32($fp)
388      li      $v0,-1                # 0xffffffffffffffff
389      bne     $v1,$v0,$L37
390      la      $a0,___sF+176
391      la      $a1,$LC20
392      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,
539             args= 24, extra= 8
540         .mask    0xd0000000,-8
541         .fmask   0x00000000,0
542         .set     noreorder
543         .cpld    $t9
544         .set     reorder
545         subu     $sp,$sp,224
546         .cprestore 24
547         sw       $ra,216($sp)
548         sw       $fp,212($sp)
549         sw       $gp,208($sp)
550         move     $fp,$sp
551         sw       $a0,224($fp)
552         sw       $a1,228($fp)
553         lw       $v1,224($fp)
554         li       $v0,1                # 0x1
555         bne      $v1,$v0,$L44
556         move     $a0,$zero
557         li       $a1,1                # 0x1
558         li       $a2,1                # 0x1
559         la       $t9,executeWithDefaultParameter
560         jal      $ra,$t9
561         sw       $v0,196($fp)
562         b        $L43
563 $L44:
564         sw       $zero,32($fp)
565         sw       $zero,36($fp)
566         sw       $zero,40($fp)
567         sw       $zero,44($fp)
568         la       $v0,$LC21
569         sw       $v0,48($fp)
570         addu     $v0,$fp,56
571         la       $v1,$LC28
572         move     $a0,$v0
573         move     $a1,$v1
574         li       $a2,112              # 0x70
575         la       $t9,memcpy
576         jal      $ra,$t9
577         sw       $zero,168($fp)
578         sw       $zero,172($fp)
579         sw       $zero,176($fp)
580         sw       $zero,180($fp)
581         sb       $zero,184($fp)
582 $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      .cpload $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
```

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 palindrome.S
```

-Wall: activa los mensajes de warning

-o: indica el archivo de salida.

Para obtener el código MIPS32

```
$ gcc -Wall -O0 -S -mrnames tpl.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. 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.

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>