
UBA FACULTAD DE INGENIERÍA

66.20 Organización de Computadoras

Trabajo Práctico 1 - Reentrega

2^{do} Cuatrimestre 2020

Integrantes:

Bacigalupo, Ivan	98064
ibacigaluppo@fi.uba.ar	
Carballo, Matías	93762
mcarballo@fi.uba.ar	
Marshall, Juan Patricio	95471
jmarshall@fi.uba.ar	

Índice

1. Introducción	2
2. Documentación	2
3. Compilación	2
4. Pruebas	2
4.1. Corridas de Prueba	3
5. Conclusión	3
6. Código en C	4
6.1. hash.c	4
6.2. hash.h	5
6.3. tp1.c	6
6.4. regressions.c	9
7. Código en MIPS	11
8. El código en MIPS generado por el compilador	13
9. El código de los tests de integración	30
10. Enunciado	32

1. Introducción

El trabajo práctico consistió en la elaboración de un programa escrito en lenguaje C, el cual consistía en el hasheo de cadenas de texto, a partir de archivos de entrada o de la entrada estándar. Ambas entradas de longitud arbitrario, por lo que termina siendo un procesamiento del tipo Streaming. El programa está escrito parcialmente en C, ya que la implementación de la función de hash_more fue realizada en Código Assembly, respetando la ABI de la cátedra para nuestro MIPS emulado en QEMU.

2. Documentación

El uso del programa (tp1) se compone de las siguientes opciones que le son pasadas por parámetro:

- `-h` o `--help`: muestra la ayuda.
- `-V` o `--version`: muestra la versión.
- `-i` o `--input`: recibe como parámetro un archivo de texto como entrada. En caso de que no usar esta opción, se toma como entrada la entrada estándar. Lo mismo ocurre si no se especifica un nombre de archivo de entrada.
- `-o` o `--output`: recibe como parámetro un archivo de texto como salida. En caso de que no usar esta opción, se toma como salida la salida estándar.

3. Compilación

El programa puede ser compilado ubicándose en la carpeta que contiene el código fuente tp1.c y correr el siguiente comando, dado un archivo Makefile:

```
./make tp1
```

4. Pruebas

Para las pruebas tenemos 2 flujos. El de tests de regresión provistos por la cátedra (para la función en assembly) y tests de integración del programa para testear su funcionamiento general.

Para el primero, basta con correr `make run_regressions`.

Mientras que para el segundo flujo, debemos correr el script `./tests.sh`.

El segundo script `/tests.sh` ejecuta las pruebas propias. Este archivo está diseñado para poder agregar pruebas de forma sencilla, simplemente se

debe agregar una línea en el sector de pruebas de la siguiente manera:

```
make_test <nombre><entrada de texto><salida esperada>
```

Este script crea los archivos correspondientes en la carpeta tests (dentro del directorio sobre el cual se ejecuta). Los archivos creados son de la forma:

- test-<nombre del test>_in: archivo de entrada
- test-<nombre del test>_out: archivo de salida generado por el programa
- test-<nombre del test>_expected: archivo de salida esperado

4.1. Corridas de Prueba

A continuación se muestran las corridas de prueba generadas por el script:

```
1  Compiling Source
2  Compilation Success
3  Starting Tests
4
5  Test: multiple_lines
6  Test passed
7
8  Test: empty_file
9  Test passed
10
11 Test: test_simple
12 Test passed
13
14 Test: test_simple2
15 Test passed
16
17 Test: only_letters_and_spaces
18 Test passed
19
20 Test: numbers_letters_and_spaces
21 Test passed
22
23 Test: long_line
24 Test passed
25
26 -----
27 All 7 tests passed!!!
28 -----
```

5. Conclusión

Al realizar este trabajo, comprendimos todo lo que implica simular un sistema operativo y lograr un entorno de desarrollo estable para trabajar a

nivel MIPS. Gracias a los flags de compilación vistos, pudimos obtener el código assembly de nuestro programa escrito en C. Nos resulto muy interesante que un programa no muy complejo y de pocas lineas, convertido a assembly ocupara tantas lineas. Es lógico, viendo que C es un lenguaje de un nivel mucho mas alto que assembly, y se abstrae de muchos elementos de la arquitectura de la computadora, que en assembly son más relevantes. Otra conclusion muy importante que obtuvimos, fue que el codigo assembly una vez que "te sacas el miedo" puede ser muy fuerte. Como dijimos antes, al ser de mas bajo nivel que C, tenemos que tener un monton de factores en consideracion (acceso a memoria, orden y uso del stack, responsabilidades de caller y callee, ect). Pero a su vez, tener el control y decision sobre todo esto, nos permite entender mucho mejor que es lo que realmente pasa a bajo nivel durante la ejecucion de nuestro programa, linea a linea.

Como conclusión general, llegamos a que este práctico nos sirvió como iniciación a todo un mundo de desarrollo a bajo nivel, usando codigo MIPS, que previamente pasamos bastante por alto.

6. Código en C

6.1. hash.c

```
1  #include <stdint.h>
2  #include <assert.h>
3
4  #include "hash.h"
5
6  void
7  string_hash_init(string_hash *h)
8  {
9      h->flag = STRING_HASH_INIT;
10     h->hash = 0;
11     h->size = 0;
12 }
13
14 /*
15 void
16 string_hash_more(string_hash *sh, char *str, size_t len)
17 {
18     assert(sh->flag == STRING_HASH_INIT || sh->flag ==
19             STRING_HASH_MORE);
20     if (sh->flag == STRING_HASH_INIT) {
21         sh->flag = STRING_HASH_MORE;
22         sh->hash = (*str) << 7;
23     }
24     while ((*str) != 0 && len--) {
25         sh->hash = (1000003 * sh->hash) ^ *str++;
26         sh->size++;
27     }
28 }
```

```
28  */
29  extern void string_hash_more(string_hash *sh, char *str, size_t
    len);
30
31  void
32  string_hash_done(string_hash *sh)
33  {
34      assert(sh->flag == STRING_HASH_INIT || sh->flag ==
          STRING_HASH_MORE);
35
36      if ((sh->hash ^= sh->size) == -1)
37          sh->hash = -2;
38
39      sh->flag = STRING_HASH_DONE;
40  }
41
42  int32_t
43  string_hash_value(string_hash *sh)
44  {
45      return sh->hash;
46  }
47
48  #if 0
49  static long string_hash(PyStringObject *a)
50  {
51      register Py_ssize_t len;
52      register unsigned char *p;
53      register long x;
54
55      if (a->ob_shash != -1)
56          return a->ob_shash;
57      len = Py_SIZE(a);
58      p = (unsigned char *) a->ob_sval;
59      x = *p << 7;
60      while (--len >= 0)
61          x = (1000003*x) ^ *p++;
62      x ^= Py_SIZE(a);
63      if (x == -1)
64          x = -2;
65      a->ob_shash = x;
66      return x;
67  }
68  #endif
```

6.2. hash.h

```
1  #ifndef _HASH_H_INCLUDED_
2  #define _HASH_H_INCLUDED_
3
4  #include <string.h>
5  #include <assert.h>
6
```

```
7  #define STRING_HASH_INIT 1
8  #define STRING_HASH_MORE 2
9  #define STRING_HASH_DONE 3
10
11  typedef struct {
12      int8_t flag;
13      int32_t hash;
14      size_t size;
15  } string_hash;
16
17
18  extern void string_hash_init(string_hash *);
19  extern void string_hash_more(string_hash *, char *, size_t);
20  extern void string_hash_done(string_hash *);
21  extern int32_t string_hash_value(string_hash *);
22  #endif
```

6.3. tp1.c

```
1  #include <stdio.h>
2  #include <stdint.h>
3  #include <string.h>
4  #include <assert.h>
5  #include <ctype.h>
6  #include <stdlib.h>
7  #include <ctype.h>
8  #include <unistd.h>
9  #include <getopt.h>
10 #include <errno.h>
11 #include "hash.h"
12
13
14 static void
15 print_help()
16 {
17     printf("\tUsage:\n"
18           "\t\tttp1 -h\n"
19           "\t\tttp1 -V\n"
20           "\t\tttp1 -i in_file -o out_file\n"
21           "\tOptions:\n"
22           "\t\t-V, --version\tPrint version and quit.\n"
23           "\t\t-h, --help\tPrint this information.\n"
24           "\t\t-i, --input\tSpecify input stream/file, \"-\" for stdin.\n"
25           "\t\t-o, --output\tSpecify output stream/file, \"-\" for stdout.\n"
26           "\tExamples:\n"
27           "\t\tttp1 < in.txt > out.txt\n"
28           "\t\tcat in.txt | ttp1 -i - > out.txt\n");
29 }
30
31 static void
```



```
32 print_usage() {
33     printf("tp1 -i in_file -o out_file\n");
34 }
35
36 static void
37 print_version(){
38     printf("tp1 2.0\n");
39 }
40
41 int
42 main(int argc, char * const argv[])
43 {
44
45     int opt= 0;
46
47     int help = -1;
48     int version = -1;
49     int input = -1;
50     int output =-1;
51
52     char *input_filename = NULL;
53     char *output_filename = NULL;
54
55     ssize_t read;
56     char * line = NULL;
57     size_t len = 0;
58
59     static struct option long_options[] = {
60         {"help",      no_argument,      0,  'h' },
61         {"version",   no_argument,      0,  'V' },
62         {"input",     required_argument, 0,  'i' },
63         {"output",    required_argument, 0,  'o' },
64         {0,           0,                 0,   0   },
65     };
66
67     int long_index = 0;
68
69     // evaluacion de los parametros enviados al programa
70     while ((opt = getopt_long(argc, argv, "hVui:o:",
71                                long_options, &long_index )) !=
72            -1) {
73
74         switch (opt) {
75             case 'h' :
76                 help = 0;
77                 break;
78             case 'V' :
79                 version = 0;
80                 break;
81             case 'i' :
82                 input = 0;
83                 input_filename = optarg;
84                 break;
85             case 'o' :
86                 output = 0;
```

```
85         output_filename = optarg;
86         break;
87     case '?':
88         exit(1);
89     default:
90         print_usage();
91         exit(EXIT_FAILURE);
92     }
93 }
94
95
96 // procesamiento de los parametros
97 if (help == 0) {
98     print_help();
99     exit(0);
100 }
101 else if (version == 0) {
102     print_version();
103     exit(0);
104 }
105
106 // estableciendo los archivos de entrada y salida
107 FILE *input_file = stdin;
108 FILE *output_file = stdout;
109 FILE *err_file = stderr;
110
111 // si vino un -i y el filename es distinto a - hacemos un
112 // open de lectura del archivo de input
113 if (input == 0 && strcmp(input_filename, "-") != 0){
114     input_file = fopen(input_filename, "r");
115     if (input_file == NULL) {
116         fprintf(err_file, "can't open input file, errno = %d\n", errno);
117         return 1;
118     }
119 }
120
121 // si vino un -o y el filename es distinto a - hacemos un
122 // open de escritura del archivo de output
123 if (output == 0 && strcmp(output_filename, "-") != 0){
124     output_file = fopen(output_filename, "w");
125     if (output_file == NULL) {
126         fprintf(err_file, "Can't open output file, errno = %d\n", errno);
127         return 1;
128     }
129 }
130
131 // Aca leemos una linea de input, inicializamos un hash,
132 // hashamos la linea y terminamos escribiendolo en output
133 while ((read = getline(&line, &len, input_file)) != -1) {
134     if (read == -1) {
135         int err = errno;
```

```
134         if (feof(input_file) == 0)
135         {
136             fprintf(err_file, strerror(err));
137             fclose(input_file);
138             fclose(output_file);
139             fclose(err_file);
140             return err;
141         }
142     }
143
144     string_hash hash;
145
146     string_hash_init(&hash);
147
148     string_hash_more(&hash, line, read);
149
150     string_hash_done(&hash);
151
152     fprintf(output_file, "0x%04x %s", string_hash_value(&
153         hash), line);
154 }
155
156 fclose(input_file);
157 fclose(output_file);
158 fclose(err_file);
159
160 return 0;
161 }
```

6.4. regressions.c

```
1  #include <stdio.h>
2  #include <stdint.h>
3  #include <string.h>
4  #include <assert.h>
5
6  #include "hash.h"
7
8  typedef struct {
9      int32_t hash;
10     char *msg;
11 } regression;
12
13 int32_t
14 get_hash_(string_hash *sh, char *msg, size_t len, size_t stride
15 )
16 {
17     char *ptr = msg;
18     size_t delta;
19     size_t rem;
```

```
20     string_hash_init(sh);
21     for (rem = len; rem > 0; ) {
22         if (rem >= stride)
23             delta = stride;
24         else
25             delta = rem;
26
27         string_hash_more(sh, ptr, delta);
28         rem -= delta;
29         ptr += delta;
30     }
31     string_hash_done(sh);
32
33     return string_hash_value(sh);
34 }
35
36 int
37 get_hash(char *msg)
38 {
39     size_t len = strlen(msg);
40     size_t stride;
41     string_hash sh;
42     int32_t h0;
43     int32_t h;
44
45     if (len > 1) {
46         h0 = get_hash_(&sh, msg, len, len);
47
48         for (stride = len; stride >= 1; stride--) {
49             h = get_hash_(&sh, msg, len, stride);
50             assert(h0 == h);
51         }
52     }
53
54     return h0;
55 }
56
57 int
58 main(int argc, char * const argv[])
59 {
60     regression regressions[] = {
61         { 0xcc2b6c5a, "66.20 Organizacion de Computadoras\n" },
62         { 0xcb5af1f1, "TP 1 - Segundo Cuatrimestre, 2020\n" },
63         { 0xcb5af1f1, "\n" },
64         { 0xd788c5a5, "Archivo de prueba TP 1.\n" },
65         { 0x91ff4b5b, "1\n" },
66         { -1, NULL },
67     };
68     regression *iter;
69     int32_t hash;
70
71     for (iter = regressions; iter->msg != NULL; iter++) {
72         hash = get_hash(iter->msg);
73         printf("0x%08x %s", hash, iter->msg);
```

```
74         assert(iter->hash == hash);  
75     }  
76  
77     return 0;  
78 }
```

7. Código en MIPS

```
1  #include <sys/syscall.h>  
2  #include <sys/regdef.h>  
3  .rdata  
4  error_msg: .asciiz "Assertion HASH_INIT or HASH_MORE failed.\n  
5               nAborted\n"  
6  .text  
7  .ent string_hash_more  
8  .globl string_hash_more  
9  
10 #Como soy Leaf no tengo A B A propio ni guardo ra  
11 #En ete caso no tengo variables adicionales por lo que L T A  
12     esta vacio  
13 # ( A B A Caller)  
14 #  a2  
15 #  a1  
16 #  a0  
17 #---S R A---  
18 #  fp  
19 #  gp  
20  
21 string_hash_more:  
22     #Stack  
23     .frame fp,8,ra  
24     subu sp,sp,8  
25  
26     #(S R A)  
27     .cprestore 0  
28     sw fp,4(sp)  
29     move fp,sp  
30  
31     #(A B A Caller)  
32     sw a2,16(sp)  
33     sw a1,12(sp)  
34     sw a0,8(sp)  
35  
36     #Assert para verificar que el hash este inicializado y que  
37     tenga el flag de more  
38     lb t0,0(a0)  
39     beq t0,1,assert_right#constante string_hash_init  
40     beq t0,2,assert_right#constante string_hash_more  
41     b assert_wrong  
42  
43 assert_right:
```

```
41
42 #If flag == hash_init
43 lb t0,0(a0)
44 li t1,1#constante string_hash_more
45 bne t1,t0,while
46
47 #flag
48 li t0,2#constante string_hash_more
49 sb t0,0(a0)
50 #hash
51 lb t0,0(a1)
52 sll t0,t0,7
53 sw t0,4(a0)
54
55
56 while:
57     #*str != 0
58     lw t3,12(sp)
59     lb t3,0(t3)
60     beqz t3,return
61     #len != 0
62     lw t3,16(sp)
63     beqz t3,return
64     addi t3,t3,-1
65     sw t3,16(sp)
66
67     #calculo de hash
68     lw t3,8(sp)
69     lw t0,4(t3)
70     mul t0,t0,1000003
71     lw t1,12(sp)# vuelvo a leer str de arriba para evitar bugs
72     lb t2,0(t1)*str
73     xor t0,t0,t2
74     sw t0,4(t3)
75     addi t1,t1,1#str++
76     sw t1,12(sp)#vuelvo a guardar a1++
77
78     #size++
79     lw t0,8(t3)
80     addi t0,t0,1
81     sw t0,8(t3)
82     b while
83 assert_wrong:
84     li a0,2
85     li a2,53
86     la a1,error_msg
87     li v0,SYS_write
88     syscall
89     li v0,SYS_exit#
90     syscall
91 return:
92     lw fp,4(sp)
93     addu sp,sp,8
94
```

```
95     jr    ra
96     .end  string_hash_more
```

8. El código en MIPS generado por el compilador

```
1      .file 1 "tp1.c"
2      .section .mdebug.abi32
3      .previous
4      .nan    legacy
5      .module fp=xx
6      .module nooddspreg
7      .abicalls
8      .rdata
9      .align 2
10     $LC0:
11     .ascii "\011Usage:\012\011\011tp1 -h\012\011\011tp1 -V
12           \012\011\011"
13     .ascii "tp1 -i in_file -o out_file\012\011Options
14           :\012\011\011-V"
15     .ascii ", --version\011Print version and quit
16           .\012\011\011-h, --"
17     .ascii "help\011Print this information.\012\011\011-i,
18           --input\011"
19     .ascii "Specify input stream/file, \"-\" for stdin
20           .\012\011\011-"
21     .ascii "o, --output\011Specify output stream/file,
22           \"-\" for std"
23     .ascii "out.\012\011Examples:\012\011\011tp1 < in.txt
24           > out.txt\012"
25     .ascii "\011\011cat in.txt | tp1 -i - > out.txt\000"
26     .text
27     .align 2
28     .set    nomips16
29     .set    nomicromips
30     .ent    print_help
31     .type   print_help, @function
32     print_help:
33     .frame  $fp,32,$31          # vars= 0, regs= 2/0,
34           args= 16, gp= 8
35     .mask   0xc0000000,-4
36     .fmask  0x00000000,0
37     .set    noreorder
38     .cpload $25
39     .set    nomacro
40     addiu   $sp,$sp,-32
41     sw      $31,28($sp)
42     sw      $fp,24($sp)
43     move    $fp,$sp
44     .cprestore 16
45     lw      $2,%got($LC0)(28)
46     addiu   $4,$2,%lo($LC0)
```

```

39      lw      $2,%call16(puts)($28)
40      move    $25,$2
41      .reloc   1f,R_MIPS_JALR,puts
42  1:      jalr   $25
43      nop
44
45      lw      $28,16($fp)
46      nop
47      move    $sp,$fp
48      lw      $31,28($sp)
49      lw      $fp,24($sp)
50      addiu   $sp,$sp,32
51      jr      $31
52      nop
53
54      .set     macro
55      .set     reorder
56      .end     print_help
57      .size    print_help, .-print_help
58      .rdata
59      .align   2
60  $LC1:
61      .ascii   "tp1 -i in_file -o out_file\000"
62      .text
63      .align   2
64      .set     nomips16
65      .set     nomicromips
66      .ent     print_usage
67      .type    print_usage, @function
68  print_usage:
69      .frame    $fp,32,$31          # vars= 0, regs= 2/0,
        args= 16, gp= 8
70      .mask     0xc0000000,-4
71      .fmask    0x00000000,0
72      .set     noreorder
73      .cpload   $25
74      .set     nomacro
75      addiu    $sp,$sp,-32
76      sw       $31,28($sp)
77      sw       $fp,24($sp)
78      move     $fp,$sp
79      .cprestore 16
80      lw       $2,%got($LC1)($28)
81      addiu    $4,$2,%lo($LC1)
82      lw       $2,%call16(puts)($28)
83      move     $25,$2
84      .reloc   1f,R_MIPS_JALR,puts
85  1:      jalr   $25
86      nop
87
88      lw      $28,16($fp)
89      nop
90      move    $sp,$fp
91      lw      $31,28($sp)

```



```

92         lw      $fp,24($sp)
93         addiu   $sp,$sp,32
94         jr      $31
95         nop
96
97         .set     macro
98         .set     reorder
99         .end     print_usage
100        .size    print_usage, .-print_usage
101        .rdata
102        .align   2
103$LC2:
104        .ascii   "tp1 2.0\000"
105        .text
106        .align   2
107        .set     nomips16
108        .set     nomicromips
109        .ent     print_version
110        .type    print_version, @function
111print_version:
112        .frame    $fp,32,$31          # vars= 0, regs= 2/0,
113                args= 16, gp= 8
114        .mask     0xc0000000,-4
115        .fmask    0x00000000,0
116        .set     noreorder
117        .cpload   $25
118        .set     nomacro
119        addiu     $sp,$sp,-32
120        sw        $31,28($sp)
121        sw        $fp,24($sp)
122        move      $fp,$sp
123        .cprestore 16
124        lw        $2,%got($LC2)($28)
125        addiu     $4,$2,%lo($LC2)
126        lw        $2,%call16(puts)($28)
127        move      $25,$2
128        .reloc    1f,R_MIPS_JALR,puts
1291:        jalr     $25
130        nop
131
132        lw        $28,16($fp)
133        nop
134        move      $sp,$fp
135        lw        $31,28($sp)
136        lw        $fp,24($sp)
137        addiu     $sp,$sp,32
138        jr        $31
139        nop
140
141        .set     macro
142        .set     reorder
143        .end     print_version
144        .size    print_version, .-print_version
145        .rdata

```

```

145      .align 2
146 $LC3:
147      .ascii "hVui:o:\000"
148      .align 2
149 $LC4:
150      .ascii "-\000"
151      .align 2
152 $LC5:
153      .ascii "r\000"
154      .align 2
155 $LC6:
156      .ascii "can't open input file, errno = %d\012\000"
157      .align 2
158 $LC7:
159      .ascii "w\000"
160      .align 2
161 $LC8:
162      .ascii "Can't open output file, errno = %d\012\000"
163      .align 2
164 $LC9:
165      .ascii "0x%04x %s\000"
166      .text
167      .align 2
168      .globl main
169      .set nomips16
170      .set nomicromips
171      .ent main
172      .type main, @function
173 main:
174      .frame $fp,104,$31          # vars= 64, regs= 2/0,
175          args= 24, gp= 8
176      .mask 0xc0000000,-4
177      .fmask 0x00000000,0
178      .set noreorder
179      .cpload $25
180      .set nomacro
181      addiu $sp,$sp,-104
182      sw $31,100($sp)
183      sw $fp,96($sp)
184      move $fp,$sp
185      .cprestore 24
186      sw $4,104($fp)
187      sw $5,108($fp)
188      sw $0,64($fp)
189      li $2,-1                    # 0xffffffffffffffff
190      sw $2,32($fp)
191      li $2,-1                    # 0xffffffffffffffff
192      sw $2,36($fp)
193      li $2,-1                    # 0xffffffffffffffff
194      sw $2,40($fp)
195      li $2,-1                    # 0xffffffffffffffff
196      sw $2,44($fp)
197      sw $0,48($fp)
198      sw $0,52($fp)

```

```
198      sw      $0,72($fp)
199      sw      $0,76($fp)
200      sw      $0,80($fp)
201      b        $L5
202      nop
203
204 $L13:
205      lw      $2,64($fp)
206      addiu   $2,$2,-63
207      sltu    $3,$2,49
208      beq     $3,$0,$L6
209      nop
210
211      sll     $3,$2,2
212      lw      $2,%got($L8)($28)
213      addiu   $2,$2,%lo($L8)
214      addu    $2,$3,$2
215      lw      $2,0($2)
216      addu    $2,$2,$28
217      jr      $2
218      nop
219
220      .rdata
221      .align  2
222      .align  2
223 $L8:
224      .gpword $L7
225      .gpword $L6
226      .gpword $L6
227      .gpword $L6
228      .gpword $L6
229      .gpword $L6
230      .gpword $L6
231      .gpword $L6
232      .gpword $L6
233      .gpword $L6
234      .gpword $L6
235      .gpword $L6
236      .gpword $L6
237      .gpword $L6
238      .gpword $L6
239      .gpword $L6
240      .gpword $L6
241      .gpword $L6
242      .gpword $L6
243      .gpword $L6
244      .gpword $L6
245      .gpword $L6
246      .gpword $L6
247      .gpword $L9
248      .gpword $L6
249      .gpword $L6
250      .gpword $L6
251      .gpword $L6
```

```

252      .gpword $L6
253      .gpword $L6
254      .gpword $L6
255      .gpword $L6
256      .gpword $L6
257      .gpword $L6
258      .gpword $L6
259      .gpword $L6
260      .gpword $L6
261      .gpword $L6
262      .gpword $L6
263      .gpword $L6
264      .gpword $L6
265      .gpword $L10
266      .gpword $L11
267      .gpword $L6
268      .gpword $L6
269      .gpword $L6
270      .gpword $L6
271      .gpword $L6
272      .gpword $L12
273      .text
274 $L10:
275      sw      $0,32($fp)
276      b       $L5
277      nop
278
279 $L9:
280      sw      $0,36($fp)
281      b       $L5
282      nop
283
284 $L11:
285      sw      $0,40($fp)
286      lw      $2,%got(optarg)($28)
287      lw      $2,0($2)
288      sw      $2,48($fp)
289      b       $L5
290      nop
291
292 $L12:
293      sw      $0,44($fp)
294      lw      $2,%got(optarg)($28)
295      lw      $2,0($2)
296      sw      $2,52($fp)
297      b       $L5
298      nop
299
300 $L7:
301      li      $4,1                      # 0x1
302      lw      $2,%call16(exit)($28)
303      move    $25,$2
304      .reloc   1f,R_MIPS_JALR,exit
305 1:      jalr   $25

```

```

306         nop
307
308 $L6:
309         lw      $2,%got(print_usage)($28)
310         addiu   $2,$2,%lo(print_usage)
311         move    $25,$2
312         .reloc  1f,R_MIPS_JALR,print_usage
313 1:       jalr   $25
314         nop
315
316         lw      $28,24($fp)
317         li      $4,1                                # 0x1
318         lw      $2,%call16(exit)($28)
319         move    $25,$2
320         .reloc  1f,R_MIPS_JALR,exit
321 1:       jalr   $25
322         nop
323
324 $L5:
325         addiu   $2,$fp,80
326         sw      $2,16($sp)
327         lw      $2,%got(long_options.3323)($28)
328         addiu   $7,$2,%lo(long_options.3323)
329         lw      $2,%got($LC3)($28)
330         addiu   $6,$2,%lo($LC3)
331         lw      $5,108($fp)
332         lw      $4,104($fp)
333         lw      $2,%call16(getopt_long)($28)
334         move    $25,$2
335         .reloc  1f,R_MIPS_JALR,getopt_long
336 1:       jalr   $25
337         nop
338
339         lw      $28,24($fp)
340         sw      $2,64($fp)
341         lw      $3,64($fp)
342         li      $2,-1                                # 0xffffffffffffffff
343         bne     $3,$2,$L13
344         nop
345
346         lw      $2,32($fp)
347         bne     $2,$0,$L14
348         nop
349
350         lw      $2,%got(print_help)($28)
351         addiu   $2,$2,%lo(print_help)
352         move    $25,$2
353         .reloc  1f,R_MIPS_JALR,print_help
354 1:       jalr   $25
355         nop
356
357         lw      $28,24($fp)
358         move    $4,$0
359         lw      $2,%call16(exit)($28)

```

```

360      move    $25,$2
361      .reloc   1f,R_MIPS_JALR,exit
362 1:      jalr   $25
363      nop
364
365 $L14:
366      lw      $2,36($fp)
367      bne     $2,$0,$L15
368      nop
369
370      lw      $2,%got(print_version)($28)
371      addiu   $2,$2,%lo(print_version)
372      move    $25,$2
373      .reloc   1f,R_MIPS_JALR,print_version
374 1:      jalr   $25
375      nop
376
377      lw      $28,24($fp)
378      move    $4,$0
379      lw      $2,%call16(exit)($28)
380      move    $25,$2
381      .reloc   1f,R_MIPS_JALR,exit
382 1:      jalr   $25
383      nop
384
385 $L15:
386      lw      $2,%got(stdin)($28)
387      lw      $2,0($2)
388      sw      $2,56($fp)
389      lw      $2,%got(stdout)($28)
390      lw      $2,0($2)
391      sw      $2,60($fp)
392      lw      $2,40($fp)
393      bne     $2,$0,$L16
394      nop
395
396      lw      $2,%got($LC4)($28)
397      addiu   $5,$2,%lo($LC4)
398      lw      $4,48($fp)
399      lw      $2,%call16(strcmp)($28)
400      move    $25,$2
401      .reloc   1f,R_MIPS_JALR,strcmp
402 1:      jalr   $25
403      nop
404
405      lw      $28,24($fp)
406      beq     $2,$0,$L16
407      nop
408
409      lw      $2,%got($LC5)($28)
410      addiu   $5,$2,%lo($LC5)
411      lw      $4,48($fp)
412      lw      $2,%call16(fopen)($28)
413      move    $25,$2

```

```

414      .reloc    1f,R_MIPS_JALR,fopen
415 1:      jalr     $25
416      nop
417
418      lw        $28,24($fp)
419      sw        $2,56($fp)
420      lw        $2,56($fp)
421      bne       $2,$0,$L16
422      nop
423
424      lw        $2,%call16(__errno_location)($28)
425      move      $25,$2
426      .reloc    1f,R_MIPS_JALR,__errno_location
427 1:      jalr     $25
428      nop
429
430      lw        $28,24($fp)
431      lw        $2,0($2)
432      move      $5,$2
433      lw        $2,%got($LC6)($28)
434      addiu     $4,$2,%lo($LC6)
435      lw        $2,%call16(sprintf)($28)
436      move      $25,$2
437      .reloc    1f,R_MIPS_JALR,sprintf
438 1:      jalr     $25
439      nop
440
441      lw        $28,24($fp)
442      li        $2,1                # 0x1
443      b         $L21
444      nop
445
446 $L16:
447      lw        $2,44($fp)
448      bne       $2,$0,$L19
449      nop
450
451      lw        $2,%got($LC4)($28)
452      addiu     $5,$2,%lo($LC4)
453      lw        $4,52($fp)
454      lw        $2,%call16(strcmp)($28)
455      move      $25,$2
456      .reloc    1f,R_MIPS_JALR,strcmp
457 1:      jalr     $25
458      nop
459
460      lw        $28,24($fp)
461      beq       $2,$0,$L19
462      nop
463
464      lw        $2,%got($LC7)($28)
465      addiu     $5,$2,%lo($LC7)
466      lw        $4,52($fp)
467      lw        $2,%call16(fopen)($28)

```

```

468      move    $25,$2
469      .reloc   1f,R_MIPS_JALR,fopen
470 1:      jalr    $25
471      nop
472
473      lw      $28,24($fp)
474      sw      $2,60($fp)
475      lw      $2,60($fp)
476      bne     $2,$0,$L19
477      nop
478
479      lw      $2,%call16(__errno_location)($28)
480      move    $25,$2
481      .reloc   1f,R_MIPS_JALR,__errno_location
482 1:      jalr    $25
483      nop
484
485      lw      $28,24($fp)
486      lw      $2,0($2)
487      move    $5,$2
488      lw      $2,%got($LC8)($28)
489      addiu   $4,$2,%lo($LC8)
490      lw      $2,%call16(sprintf)($28)
491      move    $25,$2
492      .reloc   1f,R_MIPS_JALR,sprintf
493 1:      jalr    $25
494      nop
495
496      lw      $28,24($fp)
497      li      $2,1                      # 0x1
498      b       $L21
499      nop
500
501 $L20:
502      addiu   $2,$fp,84
503      move    $4,$2
504      lw      $2,%call16(string_hash_init)($28)
505      move    $25,$2
506      .reloc   1f,R_MIPS_JALR,string_hash_init
507 1:      jalr    $25
508      nop
509
510      lw      $28,24($fp)
511      lw      $3,72($fp)
512      lw      $4,68($fp)
513      addiu   $2,$fp,84
514      move    $6,$4
515      move    $5,$3
516      move    $4,$2
517      lw      $2,%call16(string_hash_more)($28)
518      move    $25,$2
519      .reloc   1f,R_MIPS_JALR,string_hash_more
520 1:      jalr    $25
521      nop

```



```

522
523         lw      $28,24($fp)
524         addiu   $2,$fp,84
525         move    $4,$2
526         lw      $2,%call16(string_hash_done)($28)
527         move    $25,$2
528         .reloc  1f,R_MIPS_JALR,string_hash_done
529 1:         jalr   $25
530         nop
531
532         lw      $28,24($fp)
533         addiu   $2,$fp,84
534         move    $4,$2
535         lw      $2,%call16(string_hash_value)($28)
536         move    $25,$2
537         .reloc  1f,R_MIPS_JALR,string_hash_value
538 1:         jalr   $25
539         nop
540
541         lw      $28,24($fp)
542         move    $3,$2
543         lw      $2,72($fp)
544         move    $7,$2
545         move    $6,$3
546         lw      $2,%got($LC9)($28)
547         addiu   $5,$2,%lo($LC9)
548         lw      $4,60($fp)
549         lw      $2,%call16(fprintf)($28)
550         move    $25,$2
551         .reloc  1f,R_MIPS_JALR,fprintf
552 1:         jalr   $25
553         nop
554
555         lw      $28,24($fp)
556 $L19:
557         addiu   $3,$fp,76
558         addiu   $2,$fp,72
559         lw      $6,56($fp)
560         move    $5,$3
561         move    $4,$2
562         lw      $2,%call16(getline)($28)
563         move    $25,$2
564         .reloc  1f,R_MIPS_JALR,getline
565 1:         jalr   $25
566         nop
567
568         lw      $28,24($fp)
569         sw      $2,68($fp)
570         lw      $3,68($fp)
571         li      $2,-1                                # 0xffffffffffffffff
572         bne     $3,$2,$L20
573         nop
574
575         lw      $4,56($fp)

```

```

576      lw      $2,%call16(fclose)($28)
577      move    $25,$2
578      .reloc   1f,R_MIPS_JALR,fclose
579 1:      jalr   $25
580      nop
581
582      lw      $28,24($fp)
583      lw      $4,60($fp)
584      lw      $2,%call16(fclose)($28)
585      move    $25,$2
586      .reloc   1f,R_MIPS_JALR,fclose
587 1:      jalr   $25
588      nop
589
590      lw      $28,24($fp)
591      move    $2,$0
592 $L21:
593      move    $sp,$fp
594      lw      $31,100($sp)
595      lw      $fp,96($sp)
596      addiu   $sp,$sp,104
597      jr      $31
598      nop
599
600      .set     macro
601      .set     reorder
602      .end     main
603      .size    main, .-main
604      .rdata
605      .align   2
606 $LC10:
607      .ascii   "help\000"
608      .align   2
609 $LC11:
610      .ascii   "version\000"
611      .align   2
612 $LC12:
613      .ascii   "input\000"
614      .align   2
615 $LC13:
616      .ascii   "output\000"
617      .section .data.rel.local,"aw",@progbits
618      .align   2
619      .type    long_options.3323, @object
620      .size    long_options.3323, 80
621 long_options.3323:
622      .word    $LC10
623      .word    0
624      .word    0
625      .word    104
626      .word    $LC11
627      .word    0
628      .word    0
629      .word    86

```

```

630      .word    $LC12
631      .word    1
632      .word    0
633      .word    105
634      .word    $LC13
635      .word    1
636      .word    0
637      .word    111
638      .word    0
639      .word    0
640      .word    0
641      .word    0
642      .ident   "GCC: (Debian 6.3.0-18+deb9u1) 6.3.0 20170516"

```

```

1      .file     1 "hash.c"
2      .section .mdebug.abi32
3      .previous
4      .nan      legacy
5      .module   fp=xx
6      .module   nooddspreg
7      .abicalls
8      .text
9      .align    2
10     .globl    string_hash_init
11     .set      nomips16
12     .set      nomicromips
13     .ent      string_hash_init
14     .type     string_hash_init, @function
15 string_hash_init:
16     .frame     $fp,8,$31           # vars= 0, regs= 1/0,
17         args= 0, gp= 0
18     .mask      0x40000000,-4
19     .fmask     0x00000000,0
20     .set      noreorder
21     .set      nomacro
22     addiu     $sp,$sp,-8
23     sw        $fp,4($sp)
24     move      $fp,$sp
25     sw        $4,8($fp)
26     lw        $2,8($fp)
27     li        $3,1                # 0x1
28     sb        $3,0($2)
29     lw        $2,8($fp)
30     sw        $0,4($2)
31     lw        $2,8($fp)
32     sw        $0,8($2)
33     nop
34     move      $sp,$fp
35     lw        $fp,4($sp)
36     addiu     $sp,$sp,8
37     jr        $31
38     nop
39     .set      macro

```

```

40      .set      reorder
41      .end      string_hash_init
42      .size     string_hash_init, .-string_hash_init
43      .rdata
44      .align    2
45 $LC0:
46      .ascii    "hash.c\000"
47      .align    2
48 $LC1:
49      .ascii    "sh->flag == STRING_HASH_INIT || sh->flag ==
50                STRING_HASH_"
51      .ascii    "MORE\000"
52      .text
53      .align    2
54      .globl    string_hash_more
55      .set      nomips16
56      .set      nomicromips
57      .ent      string_hash_more
58      .type     string_hash_more, @function
59 string_hash_more:
60      .frame    $fp,32,$31                # vars= 0, regs= 2/0,
61      .mask     args= 16, gp= 8
62      .mask     0xc0000000,-4
63      .fmask    0x00000000,0
64      .set      noreorder
65      .cpload   $25
66      .set      nomacro
67      addiu     $sp,$sp,-32
68      sw        $31,28($sp)
69      sw        $fp,24($sp)
70      move      $fp,$sp
71      .cprestore 16
72      sw        $4,32($fp)
73      sw        $5,36($fp)
74      sw        $6,40($fp)
75      lw        $2,32($fp)
76      lb        $3,0($2)
77      li        $2,1                      # 0x1
78      beq       $3,$2,$L3
79      nop
80      lw        $2,32($fp)
81      lb        $3,0($2)
82      li        $2,2                      # 0x2
83      beq       $3,$2,$L3
84      nop
85      lw        $2,%got(__PRETTY_FUNCTION__ .1638)($28)
86      addiu     $7,$2,%lo(__PRETTY_FUNCTION__ .1638)
87      li        $6,17                     # 0x11
88      lw        $2,%got($LC0)($28)
89      addiu     $5,$2,%lo($LC0)
90      lw        $2,%got($LC1)($28)
91      addiu     $4,$2,%lo($LC1)

```

```

92      lw      $2,%call16(__assert_fail)($28)
93      move    $25,$2
94      .reloc   1f,R_MIPS_JALR,__assert_fail
95  1:      jalr   $25
96      nop
97
98  $L3:
99      lw      $2,32($fp)
100     lb      $3,0($2)
101     li      $2,1                      # 0x1
102     bne     $3,$2,$L5
103     nop
104
105     lw      $2,32($fp)
106     li      $3,2                      # 0x2
107     sb      $3,0($2)
108     lw      $2,36($fp)
109     lb      $2,0($2)
110     sll     $3,$2,7
111     lw      $2,32($fp)
112     sw      $3,4($2)
113     b       $L5
114     nop
115
116  $L7:
117     lw      $2,32($fp)
118     lw      $3,4($2)
119     li      $2,983040                 # 0xf0000
120     ori     $2,$2,0x4243
121     mul     $3,$3,$2
122     lw      $2,36($fp)
123     addiu   $4,$2,1
124     sw      $4,36($fp)
125     lb      $2,0($2)
126     xor     $3,$3,$2
127     lw      $2,32($fp)
128     sw      $3,4($2)
129     lw      $2,32($fp)
130     lw      $2,8($2)
131     addiu   $3,$2,1
132     lw      $2,32($fp)
133     sw      $3,8($2)
134  $L5:
135     lw      $2,36($fp)
136     lb      $2,0($2)
137     beq     $2,$0,$L8
138     nop
139
140     lw      $2,40($fp)
141     addiu   $3,$2,-1
142     sw      $3,40($fp)
143     bne     $2,$0,$L7
144     nop
145

```

```
146 $L8:
147     nop
148     move    $sp,$fp
149     lw      $31,28($sp)
150     lw      $fp,24($sp)
151     addiu   $sp,$sp,32
152     jr      $31
153     nop
154
155     .set    macro
156     .set    reorder
157     .end    string_hash_more
158     .size   string_hash_more, .-string_hash_more
159     .align  2
160     .globl  string_hash_done
161     .set    nomips16
162     .set    nomicromips
163     .ent    string_hash_done
164     .type   string_hash_done, @function
165 string_hash_done:
166     .frame  $fp,32,$31          # vars= 0, regs= 2/0,
167         args= 16, gp= 8
168     .mask   0xc0000000,-4
169     .fmask  0x00000000,0
170     .set    noreorder
171     .cpload $25
172     .set    nomacro
173     addiu   $sp,$sp,-32
174     sw      $31,28($sp)
175     sw      $fp,24($sp)
176     move    $fp,$sp
177     .cprestore 16
178     sw      $4,32($fp)
179     lw      $2,32($fp)
180     lb      $3,0($2)
181     li      $2,1                # 0x1
182     beq     $3,$2,$L10
183     nop
184
185     lw      $2,32($fp)
186     lb      $3,0($2)
187     li      $2,2                # 0x2
188     beq     $3,$2,$L10
189     nop
190
191     lw      $2,%got(__PRETTY_FUNCTION__ .1645)($28)
192     addiu   $7,$2,%lo(__PRETTY_FUNCTION__ .1645)
193     li      $6,33                # 0x21
194     lw      $2,%got($LC0)($28)
195     addiu   $5,$2,%lo($LC0)
196     lw      $2,%got($LC1)($28)
197     addiu   $4,$2,%lo($LC1)
198     lw      $2,%call16(__assert_fail)($28)
199     move    $25,$2
```

```

199      .reloc    1f,R_MIPS_JALR, __assert_fail
200 1:      jalr     $25
201      nop
202
203 $L10:
204      lw        $2,32($fp)
205      lw        $2,4($2)
206      move      $3,$2
207      lw        $2,32($fp)
208      lw        $2,8($2)
209      xor       $2,$3,$2
210      move      $3,$2
211      lw        $2,32($fp)
212      sw        $3,4($2)
213      lw        $2,32($fp)
214      lw        $3,4($2)
215      li        $2,-1                # 0xffffffffffffffff
216      bne       $3,$2,$L11
217      nop
218
219      lw        $2,32($fp)
220      li        $3,-2                # 0xfffffffffffffffef
221      sw        $3,4($2)
222 $L11:
223      lw        $2,32($fp)
224      li        $3,3                 # 0x3
225      sb        $3,0($2)
226      nop
227      move      $sp,$fp
228      lw        $31,28($sp)
229      lw        $fp,24($sp)
230      addiu     $sp,$sp,32
231      jr        $31
232      nop
233
234      .set      macro
235      .set      reorder
236      .end      string_hash_done
237      .size     string_hash_done, .-string_hash_done
238      .align    2
239      .globl    string_hash_value
240      .set      nomips16
241      .set      nomicromips
242      .ent      string_hash_value
243      .type     string_hash_value, @function
244 string_hash_value:
245      .frame    $fp,8,$31            # vars= 0, regs= 1/0,
246          args= 0, gp= 0
247      .mask     0x40000000,-4
248      .fmask    0x00000000,0
249      .set      noreorder
250      .set      nomacro
251      addiu     $sp,$sp,-8
252      sw        $fp,4($sp)

```

```
252     move    $fp,$sp
253     sw      $4,8($fp)
254     lw      $2,8($fp)
255     lw      $2,4($2)
256     move    $sp,$fp
257     lw      $fp,4($sp)
258     addiu   $sp,$sp,8
259     jr      $31
260     nop
261
262     .set    macro
263     .set    reorder
264     .end    string_hash_value
265     .size   string_hash_value, .-string_hash_value
266     .rdata
267     .align  2
268     .type   __PRETTY_FUNCTION__ .1638, @object
269     .size   __PRETTY_FUNCTION__ .1638, 17
270 __PRETTY_FUNCTION__ .1638:
271     .ascii  "string_hash_more\000"
272     .align  2
273     .type   __PRETTY_FUNCTION__ .1645, @object
274     .size   __PRETTY_FUNCTION__ .1645, 17
275 __PRETTY_FUNCTION__ .1645:
276     .ascii  "string_hash_done\000"
277     .ident  "GCC: (Debian 6.3.0-18+deb9u1) 6.3.0 20170516"
```

9. El código de los tests de integración

```
1  #! /bin/bash
2
3  FAILED_TESTS=0
4  TOTAL_TESTS=0
5
6  test_file() {
7      let TOTAL_TESTS=$TOTAL_TESTS+1
8      if [ ! -f "$1" ]
9      then
10         echo -e "\e[31mNo file $1\e[0m"
11     fi
12     if [ ! -f "$2" ]
13     then
14         echo -e "\e[31mNo file $2\e[0m"
15     fi
16
17     DIFF=$(diff $1 $2)
18     if [ "$DIFF" != "" ]
19     then
20         let FAILED_TESTS=$FAILED_TESTS+1
21         echo -e "\e[1;31mTest failed!\e[0m"
22     else
```



```
23         echo -e "\e[1;32mTest passed\e[0m"
24     fi
25
26 }
27
28 execute_program() {
29     ./tp1 -i $1 -o $2
30 }
31
32 make_test() {
33     echo "Test: $1"
34
35     > "./tests/test-${1}_in"
36     > "./tests/test-${1}_out"
37     > "./tests/test-${1}_expected"
38
39     printf "$2" >> "./tests/test-${1}_in"
40     printf "$3" >> "./tests/test-${1}_expected"
41
42
43     execute_program "./tests/test-${1}_in" "./tests/test-${1}_out"
44     test_file "./tests/test-${1}_out" "./tests/test-${1}_expected"
45     echo
46 }
47
48 if [ ! -d tests ]
49 then
50     mkdir tests
51 fi
52
53 rm -r ./tests/*
54
55 if [ ! -f ./tests/log_test ]
56 then
57     touch ./tests/log_test
58 fi
59
60 echo Compiling Source
61 if ! gcc -Wall -o tp1 tp1.c hash.c hash.S; then
62     echo Compilation Failed
63     exit 1
64 fi
65 echo Compilation Success
66
67 echo "Starting Tests"
68 echo
69
70
71 # ----- Sector de PRUEBAS
72     -----
```

```
73 make_test multiple_lines "66.20 Organizacion de Computadoras\  
    nTP 1 - Segundo Cuatrimestre, 2020\n\nArchivo de prueba TP  
    1." "0xcc2b6c5a 66.20 Organizacion de Computadoras\  
    n0xcb5af1f1 TP 1 - Segundo Cuatrimestre, 2020\n0x4c4b4f0b \  
    n0xc651b96a Archivo de prueba TP 1."  
74 make_test empty_file "" ""  
75 make_test test_simple "holasssss" "0xb2583150 holasssss"  
76 make_test test_simple2 "Orga de compu" "0xdc0b73eb Orga de  
    compu"  
77 make_test only_letters_and_spaces "a b c d e f g h i" "0  
    x3dbc1a30 a b c d e f g h i"  
78 make_test numbers_letters_and_spaces "969420a6 528dsa 528 ab" "  
    0x81678420 969420a6 528dsa 528 ab"  
79 make_test long_line "  
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
    " "0x16490920  
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
    "  
80  
81 # Agregar pruebas aca  
82  
83 #  
    -----  
84  
85 echo -----  
86  
87 if [ $FAILED_TESTS == 0 ]  
88 then  
89     echo -e "\e[92mAll $TOTAL_TESTS tests passed!!!\e[0m"  
90 else  
91     echo -e "\e[91m Failed tests: $FAILED_TESTS from  
        $TOTAL_TESTS\e[0m"  
92 fi  
93  
94 echo -----
```

10. Enunciado

El enunciado se encuentra anexado al final de este documento.