Universidad de Buenos Aires - FIUBA
66.20 Organización de Computadoras
Trabajo práctico 1: Programación MIPS
2° cuatrimestre de 2020

Integrantes	**Padrón**	**Mail**
Ponce Antonella	100635	aponce@fi.uba.ar
Chavar Hugo	90541	hchavar@fi.uba.ar
Lescano Maier Aldana	99839	alescano@fi.uba.ar

Diseño e implementación del programa

Para la implementación de la función *void string_hash_more(string_hash *sh, char *str, size_t len)* se creó un stack frame de tamaño igual a 32 bytes: 16 destinados al ABA y 16 al SRA. En la siguiente tabla puede observarse la organización final de stack frame.

40	**a2 = len**	ABA (caller)
36	**a1 = str**	
32	**a0 = sh**	
28	////////////////////////// padding //////////////////////////	SRA
24	**ra**	
20	**fp**	
16	**gp**	
12		ABA
8		
4		
0		

La implementación de la función mencionada anteriormente se realizó en dos archivos distintos, *hashmore.S* y *myassert.S* a fines de mayor legibilidad del código. El segundo archivo contiene la implementación de la función *myassert* mientras que el primero la lógica restante.

Comandos para compilar el programa

Los comando utilizados para compilar el programa son los siguientes:
gcc -Wall -g -o tp1 hash.c tp1.c hashmore.S myassert.S

Pruebas

Al correr las regresiones definidas por el código suministrado por la práctica se obtuvo la siguiente salida:

```
# gcc -Wall -g -o tp1 hash.c hashmore.S myassert.S regressions.c
# ./tp1
0xcc2b6c5a 66.20 Organizacion de Computadoras
0xcb5af1f1 TP 1 - Segundo Cuatrimestre, 2020
0xcb5af1f1
0xd788c5a5 Archivo de prueba TP 1.
0x91ff4b5b 1
```

Al correr unas pruebas propias junto al archivo tp1.c se obtuvo la siguiente salida:

```
# gcc -Wall -g -o tp1 hash.c hashmore.S myassert.S tp1.c
# cat test.din | ./tp1

0xccd151ff Esto es una prueba
0x847802a3 La región urbana dejará el Aislamiento Social,
Preventivo
0x9021e0af y Obligatorio (ASPO) para ingresar a una etapa con
mayores habilitaciones.
0x553b04c7 Mantuvo ocho semanas de caída constante en contagios.
0x557d67f0
0x91ff4b5b 1
0x557d67f0
```

Archivo de entrada vacío:

```
# ./tp1 -o salida.txt </dev/null
# ls -l salida.txt
-rw-r--r-- 1 root root 0 Nov 17 20:56 salida.txt
```

Única línea de un caracter:

```
# echo 1 | ./tp1 -o -
0x91ff4b5b 1
```

Entrada alojada en sistema de archivos:

```
# echo 1 > entrada.txt
# ./tp1 -i entrada.txt -o -
0x91ff4b5b 1
```

Código fuente

El archivo *myassert.S* contiene la implementación de la función *myassert* de C escrita en assembly.

```
#include <sys/regdef.h>
#include <sys/syscall.h>


     .rdata

msg: .asciiz    "Assertion error (Assemby).\n"

     .text

     .abicalls
     .align     2
```

```
      .globl     myassert
      .ent  myassert
myassert:
      .frame     fp, 8, ra
      .set  noreorder
      .cpload t9
      .set reorder

      subu  sp, sp, 8
      .cprestore 0
      sw    fp, 4(sp)
      move  fp, sp

      sw    a0,8(fp)

      beq   a0, zero, myerr # si argumento es falso => error
      b     fin   # sale ok
myerr:
      #hay que mostrar error y terminar el proceso
      li    a0, 2 # stderr
      la    a1, msg    # error a mostrar
      li    a2, 27     # longitud del mensaje
      li    v0, SYS_write
      syscall

      #chequeo de error
      beq   a3, zero, salir
      li    v0, -1
      j     fin

salir:
      li    a0, 1
      li    v0, SYS_exit
      syscall


fin:
      move  sp,fp
      lw    fp,4(sp)
      addiu sp,sp,8
      jr    ra

      .end myassert
```

El archivo hashmore.S contiene la implementación de la función void string_hash_more(string_hash *, char * size_t) de C escrita en assembly.

```
#include <sys/regdef.h>
```

```
#include <sys/syscall.h>

      .text
      .abicalls
      .align 2
      .globl string_hash_more
      .ent string_hash_more

string_hash_more:

      .frame fp, 48, ra
      .set  noreorder
      .cpload t9
      .set reorder

      #stack frame
      subu  sp, sp, 48
      .cprestore 32
      sw    fp, 36(sp)
      sw    ra, 40(sp)
      move  fp, sp

      #salvo argumento
      sw    a0, 48(fp) #sh
      sw    a1, 52(fp) #str
      sw    a2, 56(fp) #len

      #armo struct
      lb    t0, 0(a0)  #flag
      lw    t1, 4(a0)  #hash
      lw    t2, 8(a0)  #size

      # myassert
      addiu t3, zero, 1
      addiu t4, zero, 2

      sne   t5, t3, t0 # flag = STRING_HASH_INIT;
      sne   t6, t4, t0 # flag = STRING_HASH_MORE;
      or    a0, t5, t6
      la    t9, myassert
      jal   t9

      lw    a0, 48(fp)
      lb    t0, 0(a0)  #flag
      addiu t3, zero, 1
      beq   t0, t3, resuelvo_if
      b     while
```

```
resuelvo_if:

      #sh->flag = STRING_HASH_MORE;
      addiu t0, zero, 2     #sh->flag = 2
      lw    a0, 48(fp)
      sb    t0, 0(a0)  #actualizo sh->flag

      #sh->hash = (*str) << 7;
      lw    t2, 52(fp) #desreferencio str
      lb    t2, 0(t2)  #levanto el primer byte
      sll   t1, t2, 7
      sw    t1, 4(a0)  #acualizo hash directo en la memoria

while:

      lw    t0, 52(fp) # desreferencio str
      lb    t2, 0(t0)  #levanto el primer byte
      beq   t2, zero, salir
      lw    a2, 56(fp)
      beq   a2, zero, salir

      # Se cumplen ambas condiciones,cuerpo del while:

      lw    a0, 48(fp)
      lw    t1, 4(a0)  #traigo sh->hash
      mul   t1, t1, 1000003 #(1000003 * sh->hash)
      xor   t1, t1, t2
      sw    t1, 4(a0)  #actualizo hash
      lw    t0, 52(fp)
      addiu t0, t0, 1  #str++
      sw    t0, 52(fp) #actualizo str

      lw    t2, 8(a0) #traigo sh->size
      addiu t2, t2, 1 #size++
      sw    t2, 8(a0) #actualizo sh->size

      addiu t1, zero, 1 #t1 = 1
      subu  a2, a2, t1 #len--
      sw    a2, 56(fp)

      b     while

salir:
      # (esta función no devuelve nada)
      lw    fp, 36(sp)
      lw    ra, 40(sp)
      addiu sp, sp, 48
      jr    ra
```

```
      .end string_hash_more
```

El archivo *tp1.c* contiene la implementación del manejo del programa escriba en C.

```
#include <errno.h>
#include <fcntl.h>
#include <getopt.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include "hash.h"

#define BUFFERSIZ 256

int si = 1; /* es standard input: por defecto es true */
int so = 1; /* es standard output: por defecto es true */

FILE *input;
FILE *output;


typedef struct {
      int32_t hash;
      char *msg;
} regression;

int32_t
get_hash_(string_hash *sh, char *msg, size_t len, size_t stride)
{
      char *ptr = msg;
      size_t delta;
      size_t rem;

      string_hash_init(sh);
      for (rem = len; rem > 0; ) {
            if (rem >= stride)
                  delta = stride;
            else
                  delta = rem;

            string_hash_more(sh, ptr, delta);
            rem -= delta;
            ptr += delta;
```

```
      }
      string_hash_done(sh);

      return string_hash_value(sh);
}

int
get_hash(char *msg)
{
      size_t len = strlen(msg);
      size_t stride;
      string_hash sh;
      int32_t h0;
      int32_t h;

      if (len > 1) {
            h0 = get_hash_(&sh, msg, len, len);

            for (stride = len; stride >= 1; stride--) {
                  h = get_hash_(&sh, msg, len, stride);
                  assert(h0 == h);
            }
      }

      return h0;
}


int
main (int argc, char **argv)
{
  int c;
  input = stdin;
  output = stdout;

  static struct option long_options[] =
  {
    {"help",    no_argument,       0, 'h'},
    {"version", no_argument,       0, 'V'},
    {"input",   required_argument, 0, 'i'},
    {"output",  required_argument, 0, 'o'},
    {0, 0, 0, 0}
  };


  while (1)
  {
    /* getopt_long stores the option index here. */
```

```
    int option_index = 0;

    c = getopt_long (argc, argv, "hV:i:o:",
                     long_options, &option_index);

    /* Detect the end of the options. */
    if (c == -1)
      break;

    switch (c)
      {

      case 'h':
        fprintf(stderr, "Usage:\n  %s -h \n  %s -v\n  %s -i
in_file -o out_file\n", argv[0], argv[0], argv[0]);
        fprintf(stderr, "Options:\n  -h, --help     Print this
information and quit.\n  -V, --version  Print version and quit.\n
-i, --input    Specify input stream/file, \"-\" for stdin.\n  -o,
--output   Specify output stream/file, \"-\" for stdout.\n");
        fprintf(stderr, "Examples:\n  tp1 < in.txt > out.txt \n
cat in.txt | tp1 -i - > out.txt\n");
        break;

      case 'V':
        puts ("tp1 orga6620 v1.0\n");
        break;

      case 'i':

        si = 0;
        if (strcmp(optarg, "-")) {

          input = fopen(optarg, "r");
          if (input == NULL)
          {
            fprintf (stderr, "Can't open file `%s` for reading.
%s\n", optarg, strerror(errno));
            exit (1);
          }

        }

        break;

      case 'o':

        so = 0;
        if (strcmp(optarg, "-")) {
```

```
          output = fopen(optarg, "w+");
          if (output == NULL)
          {
            fprintf (stderr, "Can't open file `%s` for writing. %s
\n", optarg, strerror(errno));
            exit (1);
          }

        }

        break;

      default:
        exit(1);
      }
  }

  /* No pueden haber mas argumentos se da error */
  if (optind < argc)
  {
    //printf ("non-option ARGV-elements: ");
    while (optind < argc)
      fprintf (stderr, "%s non-valid argument\n", argv[optind++]);

    fclose(input);
    fclose(output);
    exit(1);
  }

  size_t linesiz = 0;
  char* linebuf = NULL;
  ssize_t linelen = 0;
  int32_t hash;
  while ((linelen = getline(&linebuf, &linesiz, input) > 0))
  {
    hash = get_hash(linebuf);
    //process_line(linebuf, linesiz);

    char * tempStr = (char *) malloc(strlen(linebuf)+11);
    sprintf(tempStr, "0x%08x %s", hash, linebuf);
    //printf("[%s]\n", linebuf);
    fwrite(tempStr, 1, strlen(tempStr), output);
    free(tempStr);

  }

  free(linebuf);
  fclose(input);
```

```
  fclose(output);

  exit (0);
}
```

Código MIPS 32

```
tp1:     file format elf32-tradbigmips


Disassembly of section .init:

0000077c <_init>:
 77c: 3c1c0002    lui   gp,0x2
 780: 279c8b64    addiu gp,gp,-29852
 784: 0399e021    addu  gp,gp,t9
 788: 27bdffe0    addiu sp,sp,-32
 78c: afbc0010    sw    gp,16(sp)
 790: afbf001c    sw    ra,28(sp)
 794: 8f82807c    lw    v0,-32644(gp)
 798: 10400004    beqz  v0,7ac <_init+0x30>
 79c: 00000000    nop
 7a0: 8f99807c    lw    t9,-32644(gp)
 7a4: 0320f809    jalr  t9
 7a8: 00000000    nop
 7ac: 8fbf001c    lw    ra,28(sp)
 7b0: 03e00008    jr    ra
 7b4: 27bd0020    addiu sp,sp,32

Disassembly of section .text:

000007c0 <__start>:
     7c0:   03e00025    move  zero,ra
     7c4:   04110001    bal   7cc <__start+0xc>
     7c8:   00000000    nop
     7cc:   3c1c0002    lui   gp,0x2
     7d0:   279c8b14    addiu gp,gp,-29932
     7d4:   039fe021    addu  gp,gp,ra
     7d8:   0000f825    move  ra,zero
     7dc:   8f848018    lw    a0,-32744(gp)
     7e0:   8fa50000    lw    a1,0(sp)
     7e4:   27a60004    addiu a2,sp,4
     7e8:   2401fff8    li    at,-8
     7ec:   03a1e824    and   sp,sp,at
     7f0:   27bdffe0    addiu sp,sp,-32
     7f4:   8f87801c    lw    a3,-32740(gp)
     7f8:   8f888020    lw    t0,-32736(gp)
     7fc:   afa80010    sw    t0,16(sp)
     800:   afa20014    sw    v0,20(sp)
     804:   afbd0018    sw    sp,24(sp)
     808:   8f998074    lw    t9,-32652(gp)
     80c:   0320f809    jalr  t9
```

```
810:   00000000    nop
```