Facultad de Ingeniería
Universidad de Buenos Aires

Asignatura: Organización de Computadoras

TP0: Infraestructura básica

Alumnos:

Nombre	Padrón
Camila Serra	97422
Cynthia Marlene Gamarra Silva	92702
Gonzalo Almendro	80698

21 de octubre de 2020

Índice

Índice **1**

1. Enunciado del trabajo práctico **2**

2. Diseño e implementación **5**

3. Parámetros del programa **8**

4. Compilación del programa **8**

5. Pruebas realizadas **10**
5.1. Pruebas con archivo bash test-automatic.sh 10
5.1.1. Generales 11

6. Conclusiones **12**

Referencias **12**

A. Código fuente **13**
A.1. Código en C 13
A.1.1. main.c 13
A.1.2. constants.h 14
A.1.3. command.h 14
A.1.4. command.c 15
A.1.5. file.h 17
A.1.6. file.c 18
A.1.7. encode.h 19
A.1.8. encode.c 19
A.2. Código MIPS32 22
A.2.1. tp0.S 22

1. Enunciado del trabajo práctico

66:20 Organización de Computadoras
Trabajo práctico 0: Infraestructura básica

1. Objetivos

Familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementando un programa (y su correspondiente documentación) que resuelva el problema piloto que presentaremos más abajo.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes.

Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 7), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo, y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

4. Recursos

Usaremos el programa QEMU [1] para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión reciente del sistema operativo Debian [2].

5. Base 64

La codificación base 64 [3] se creó para poder transmitir archivos binarios en medios que sólo admitían texto: 64 es la mayor potencia de 2 que se podía

representar sólo con caracteres ASCII imprimibles. Básicamente se tiene una tabla de conversión de combinaciones de 6 bits a caracteres ASCII, se 'corta' el archivo en secuencias de 6 bits y se transmiten los caracteres correspondientes a esas secuencias. Cada tres bytes de la secuencia original se generan cuatro caracteres base64; cuando la cantidad de bytes original no es múltiplo de tres, se adicionan caracteres '=' al final en cantidad necesaria.

6. Programa

El programa a escribir, en lenguaje C, recibirá un nombre de archivo (o el archivo mismo por `stdin`) y devolverá ese mismo archivo codificado en `base64` [3], o bien decodificado desde base64 si se utiliza la opción `-d`.

6.1. Ejemplos

Primero, usamos la opción `-h` para ver el mensaje de ayuda:

```
$ tp0 -h
Usage:
  tp0 -h
  tp0 -V
  tp0 [options]
Options:
  -V, --version     Print version and quit.
  -h, --help        Print this information.
  -o, --output      Path to output file.
  -i, --input       Path to input file.
  -d, --decode      Decode a base64-encoded file.
Examples:

  tp0 -i input.txt -o output.txt
```

Luego, lo usamos para codificar un pequeño fragmento de texto:

```
$ cat quijote.txt
En un lugar de La Mancha de cuyo nombre no quiero acordarme
$ ./tp0 -i quijote.txt -o qb64
$ cat qb64
RW4gdW4gbHVnYXIgZGUgTGEgTWFuY2hhIGRlIGN1eW8gbm9tYnJlIG5vIHF1aWVybyBhY29yZGFy
bWUK
```

Otra manera de ejecutarlo es a través de `stdin` y/o `stdout`:

```
cat quijote.txt | ./tp0
RW4gdW4gbHVnYXIgZGUgTGEgTWFuY2hhIGRlIGN1eW8gbm9tYnJlIG5vIHF1aWVybyBhY29yZGFy
bWUK
```

También se puede usar para decodificar:

```
$ ./tp0 -d -i qb64 -o texto
$ cat texto
En un lugar de La Mancha de cuyo nombre no quiero acordarme
```

7. Informe

El informe deberá incluir:

- Documentación relevante al diseño e implementación del programa;
- Las corridas de prueba, con los comentarios pertinentes;
- El código fuente, en lenguaje C;
- El código MIPS32 generado por el compilador;
- Este enunciado.

8. Fecha de entrega

La fecha de entrega es el jueves 22 de Octubre de 2020.

Referencias

[1] QEMU, `https://www.qemu.org/`.

[2] Debian, the Universal Operating System, `https://www.debian.org/`.

[3] Codificación base64, `https://es.wikipedia.org/wiki/Base64`

2. Diseño e implementación

El programa que contiene la lógica de codificador y decodificador se encuentra en el archivo ***encode.c***. El codificador transforma expresiones con caracteres ASCII en base64, mientras que el decodificador hace el proceso inverso. Ambas funciones se implementan con diversos ciclos y recorridos, así como utilización de instrucciones aritmético lógicas (sumas, &, shifts, etc) que relacionan la codificación base 64 con ASCII.

Se creó un achivo ***file.c*** para el manejo de los archivos a codificar/decodificar. Además se creó otro archivo llamado ***command.c*** cuya función es tomar los parámetros del programa y realizar las acciones pertinentes a la misma.

Específicamente el programa se estructura en los siguientes pasos:

- Análisis de las parámetros de la línea de comandos: se analizan las opciones ingresadas por la línea de comandos utilizando la función `getopt_long()`, la cual puede procesar cada opción que es leída de forma simplificada. Se extraen los argumentos de cada opción y se los guarda dentro de una estructura para su posterior acceso del tipo `command_options_st` cuya definición es

```
1     typedef struct {
2         File input_file;
3         File output_file;
4         const char *input_path;
5         const char *output_path;
6         char encode_option;
7         char error_condition;
8     } command_options_st;
9
```

  En caso de que no se encuentre alguna opción, se muestra el mensaje de ayuda al usuario para que identifique el prototipo de cómo debe ejecutar el programa.

- Validación de opciones: a medida que se va analizando cada opción de la línea de comandos, se valida cada una de ellas. Si se ingresó algún parámetro no válido para el programa o si se encuentró un error se lo informa al usuario por pantalla y se aborta la ejecución del programa. Se utiliza para ello se la función `show_error()` cuyo resultado es:

```
1     printf("Options:\n");
2     printf("\t-V,\t--version\tPrint version and quit.\n");
3     printf("\t-h,\t--help\t\tPrint this information.\n");
4     printf("\t-i,\t--input\t\tLocation of the input file.\n");
5     printf("\t-o,\t--output\tLocation of the output file.\n");
6     printf("\t-d,\t--decode\tDecode a base64-encoded file (default is encode).\n");
7     printf("Examples:\n");
8     printf("\ttp0 -i ~/input -o ~/output\n");
9     printf("\ttp0 --decode\n");
10
```

  Para el caso en que no hubo errores a la validación de los argumentos se procede a llamar a las funciones correspondientes a:

  - `Mensaje de ayuda:` Función `show_help()`
  - `Mensaje de versión:` Función `show_version()`
  - `Input file :` Función `set_input_file()` que guarda la entrada del archivo donde será leído el texto.
  - `Output file:` Función set_output_file() que guarda la entrada del archivo de salida donde se escribirá el texto codificado.
  - `Acción del programa a ejecutar:` Si se recibe la opción de decodificar se llama a la función set_decode() que setea la variable $opt->encode\_option$ indicando que es DECODE (está seteado en default ENCODE)

- Encode/Decode: una vez que se procesó correctamente las opciones de la línea de comandos se procede a llamar a la función `process()` que ejecutará la operación de ENCODE o DECODE dependiendo del argumento pasado en la línea de comandos. La operación de DECODE se ejecuta el siguiente código:

```
    while (!file_eof(&opt->input_file) && !has_errors(opt)) {
            unsigned int read = file_read(&opt->input_file, buf_encoded, 4);
            if (read > 0) {
                if (read != 4) {
                    set_error(opt, INVALID_FILE_LENGTH);
                    show_error(opt);
                } else {
                    ++count;
                    if (count == 18) { // 19 * 4 =  76 bytes
                        unsigned char aux;
                        file_read(&opt->input_file, &aux, 1);
                        count = 0;
                    }

                    if (Decode(buf_encoded, buf_decoded)) {
                        char aux = 0;
                        if (buf_encoded[2] == '=') {
                            ++aux;
                        }
                        if (buf_encoded[3] == '=') {
                            ++aux;
                        }

                        file_write(&opt->output_file, buf_decoded, 3 - aux);
                    } else {
                        set_error(opt, INVALID_CHARS);
                        show_error(opt);
                        unsigned int i;
                        for (i = 0; i < 4; ++i) {
                            fprintf(stderr, "%c", buf_encoded[i]);
                        }
                    }
                }

            }
        }
```

Básicamente lo que se realiza es la lectura del archivo para procesarlo teniendo en cuenta la longitud del archivo a procesar y el padding a decodificar. La función `Decode()` retorna un buffer de 3 caracteres con el decode de 4 caracteres en base64. Se debe cumplir:

Pre: el buffer input contiene 4 caracteres. El buffer output tiene por lo menos 3 caracteres

Post: retorna un buffer de 3 byte con los caracteres en ASCII. retorna 0 si error 1 si ok

La operación de ENCODE se ejecuta el siguiente código:

```
        while (!file_eof(&opt->input_file)) {
            memset(buf_decoded, 0, 3);
            unsigned int read = file_read(&opt->input_file, buf_decoded, 3);
            if (read > 0) {
                Encode(buf_decoded, read, buf_encoded);
                file_write(&opt->output_file, buf_encoded, 4);
                ++count;
                if (count == 18) // 19 * 4 =  76 bytes
                {
                    file_write(&opt->output_file, (unsigned char *) "\n", 1);
                    count = 0;
                }
            }
        }
```

Básicamente lo que se realiza es la lectura del archivo para procesarlo en la función `Encode()`e n donde recibe 3 caracteres en buffer y los convierte en 4 caracteres codificados en output. Se debe cumplir:

Pre: el buffer contiene length caracteres (1 a 3) y todos los caracteres son validos

Post: retorna un buffer de 4 byte con los caracteres en base64.

3. Parámetros del programa

Se detallan a continuación los parámetros del programa

- -h: Visualiza la ayuda del programa, en la que se indican los parámetros y sus objetivos.
- -V: Indica la versión del programa.
- -i: Archivo de entrada del programa.
- -o: Archivo de salida del programa.
- -d: Indica decodificación a base64.

Se indica a continuación detalles respecto a los parámetros:

- Si no se explicitan -i y -o, se utilizarán stdin y stdout, respectivamente.
- -V es una opción "show and quit". Si se explicita este parámetro, sólo se imprimirá la versión, aunque el resto de los parámetros se hayan explicitado.
- -h también es de tipo "show and quit" y se comporta de forma similar a -V.
- en caso de que se use la entrada estándar (con comando echo texto | ./tp0 -a encode) y luego se especifique un archivo de salida con -i, prevalecerá el establecido por parámetro.

4. Compilación del programa

Para obtener un ejecutable, se creó un archivo `makefile` cuyo contenido es:

```
CC = gcc
CFLAGS = -o0 -g -Wall -Werror -pedantic -std=c99

OBJECTS = command.o encode.o file.o
EXEC = tp0

all: $(EXEC)

command.o: command.c command.h
	$(CC) $(CFLAGS) -c command.c -o command.o

encode.o: encode.c encode.h
	$(CC) $(CFLAGS) -c encode.c -o encode.o

file.o: file.c file.h
	$(CC) $(CFLAGS) -c file.c -o file.o

$(EXEC): $(OBJECTS)
	$(CC) $(CFLAGS) $(OBJECTS) main.c -o $(EXEC) -lm


run: $(EXEC)
	./$(EXEC)


clean:
	rm -f *.o $(EXEC)

```

Para ejecutarlo, posicionarse en el directorio `src/` y ejecutar el siguiente comando:

```
$ make
```

Para proceder a la ejecución del programa, se debe llamar a:

```
$ ./tp0
```

seguido de los parámetros que se desee modificar, los cuales se indicaron en la sección 1.2.

En caso de ser entrada estándar (stdin) se podrá ejecutar de la siguiente forma:

```
$ echo texto | ./tp0 -a encode
```

También en este caso, se indican a continuación los parámetros a usar.

Para el caso de hacerlo en el programa QEMU que provee la cátedra, utilizando la máquina virtual que contiene el sistema operativo Debian, no se utilizó el archivo Makefile, la compilación se realizó con la herramienta `gcc`.

5. Pruebas realizadas

5.1. Pruebas con archivo bash test-automatic.sh

Para la ejecución del siguiente script se debe copiar, se debe ubicar el archivo ejecutable compilado dentro de la carpeta de test para que se ejecuten correctamente las pruebas. El script sería:

```
#!/bin/bash

echo "############################################################"
echo "#################### Tests Automaticos #####################"
echo "############################################################"
echo ""

echo "#-----------------------------------------------------------"
echo "#----------# COMIENZA Test 01 - Archivo Vacio #-----------"

touch ./test-out/zero.txt
./tp0 -i ./test-out/zero.txt -o ./test-out/zero-encoded.txt
ls -l ./test-out/zero-encoded.txt

if diff -b ./test-out/zero.txt ./test-out/zero-encoded.txt; then
        echo "[OK]";
else
        echo "[ERROR]";
fi

echo "#----------# TERMINA  Test 01 - Archivo Vacio #-----------"
echo "#-----------------------------------------------------------"
echo ""

echo "#----------# COMIENZA Test 02 - Stdin Input   #-----------"

echo -n Test02 > ./test-out/stdin-test.txt
ls -l ./test-out/stdin-test.txt

cat ./test-out/stdin-test.txt | ./tp0 -o ./test-out/stdin-encoded.txt
ls -l ./test-out/stdin-encoded.txt

cat ./test-out/stdin-encoded.txt | ./tp0 --decode -o ./test-out/stdin-decoded.txt
ls -l ./test-out/stdin-decoded.txt

if diff -b ./test-out/stdin-test.txt ./test-out/stdin-decoded.txt; then
        echo "[OK]";
else
        echo "[ERROR]";
fi


echo "#----------# TERMINA  Test 02 - Stdin Input   #-----------"
echo "#-----------------------------------------------------------"
echo ""


echo "#----------# COMIENZA Test 03 - Stdout Output  #-----------"

echo -n Test03 > ./test-out/stdout-test.txt
ls -l ./test-out/stdout-test.txt

./tp0 -i ./test-out/stdout-test.txt > ./test-out/stdout-encoded.txt
ls -l ./test-out/stdout-encoded.txt

./tp0 --decode -i ./test-out/stdout-encoded.txt > ./test-out/stdout-decoded.txt
ls -l ./test-out/stdout-decoded.txt

if diff -b ./test-out/stdout-test.txt ./test-out/stdout-decoded.txt; then
        echo "[OK]";
```

```
else
        echo "[ERROR]";
fi


echo "#----------# TERMINA  Test 02 - Stdout Output  #-----------"
echo "#------------------------------------------------------------"
echo ""



echo "#----------# COMIENZA Test 04 - Stdin Stdout  #-----------"

echo -n Test04 > ./test-out/stdin-stdout-test.txt
ls -l ./test-out/stdin-stdout-test.txt

cat ./test-out/stdin-stdout-test.txt | ./tp0 > ./test-out/stdin-stdout-encoded.txt
ls -l ./test-out/stdin-stdout-encoded.txt

cat ./test-out/stdin-stdout-encoded.txt | ./tp0 --decode > ./test-out/stdin-stdout-decoded.txt
ls -l ./test-out/stdin-stdout-decoded.txt

if diff -b ./test-out/stdin-stdout-test.txt ./test-out/stdin-stdout-decoded.txt; then
        echo "[OK]";
else
        echo "[ERROR]";
fi


echo "#----------# TERMINA  Test 04 - Stdin Stdout  #-----------"
echo "#------------------------------------------------------------"
echo ""
```

El cual no presenta errores en ninguna de las corridas llevadas a cabo.

Todas las pruebas que se presentan a continuación, están codificadas en los archivos de prueba ***.txt de forma que puedan ejecutarse y comprobar los resultados obtenidos.

Se indicaran a continuación lo siguiente: comandos para ejecutarlas, líneas de código que las componen y resultado esperado.

5.1.1. Generales

- Mensaje de ayuda

```
$$ ./tp0 -h   o ./tp0  --help
Options:
        -V,     --version       Print version and quit.
        -h,     --help          Print this information.
        -i,     --input         Location of the input file.
        -o,     --output        Location of the output file.
        -d,     --decode        Decode a base64-encoded file (default is encode).
Examples:
        tp0 -i ~/input -o ~/output
        tp0 --decode
```

- Mensaje de version

```
$ ./tp0 -V   o ./tp0  --version
Version: 1.0

```

6. Conclusiones

El trabajo práctico nos permitió desarrollar una API para procesar archivos transformándolos a su equivalente `base64` en lenguaje C. Además, nos permitió familiarizarnos con la arquitectura MIPS32 y el consecuente análisis y desarrollo de código assembler MIPS utilizando el programa QEMU para simular el entorno de desarrollo.

Referencias

[1] Base64 (Wikipedia) http://en.wikipedia.org/wiki/Base64

[2] Script QEMU, Script provisto por la práctica/

[3] QEMU, https://www.qemu.org/

[4] Kernighan, B. W. - Ritchie, D. M. - *C Programming Language* - 2nd edition - Prentice Hall - 1988.

[5] *GNU Make* - https://www.gnu.org/software/make/

[6] *Valgrind* - http://valgrind.org/

A. Código fuente

A.1. Código en C

A.1.1. main.c

```
#include <stddef.h>
#include <getopt.h>
#include "constants.h"
#include "command.h"

int main(int argc, char **argv) {
    struct option arg_long[] = {
            {"input",   required_argument, NULL, 'i'},
            {"output",  required_argument, NULL, 'o'},
            {"decode",  no_argument, NULL, 'd'},
            {"help",    no_argument, NULL, 'h'},
            {"version", no_argument, NULL, 'V'},
    };

    char arg_opt_str[] = "i:o:dhV";
    int arg_opt;
    int arg_opt_idx = 0;

    command_options_st cmd_options;
    // Default Values: encode, stdin as input, stdout as output, stderr as error output
    command_create(&cmd_options);

    char should_process = TRUE;
    while ((arg_opt = getopt_long(argc, argv, arg_opt_str, arg_long, &arg_opt_idx)) != -1
           &&
           should_process) {

        switch (arg_opt) {
            case 'i': {
                // Set Input File
                set_input_file(&cmd_options, optarg);
            }
                break;
            case 'o': {
                // Set Output File
                set_output_file(&cmd_options, optarg);
            }
                break;
            case 'h': {
                // Show Options
                show_help();
                should_process = FALSE;
            }
                break;
            case 'V': {
                // Show Version
                show_version();
                should_process = FALSE;
            }
                break;
            case 'd': {
                // Set Decode option
                set_decode(&cmd_options);
            }
                break;
            default: {
                // Set Error Condition = INVALID_ARGUMENT
                // [Gonzalo: We analize only valid arguments values here: i, o, h, V, d]
                set_error(&cmd_options, INVALID_ARGUMENT);
            }
                break;
        }
```

```
    }

    // Help or Version arguments, no processing
    if (!should_process) {
        return 0;
    }

    // Processs Encode/Decode if no errors found, otherwise show error message
    if (has_errors(&cmd_options)) {
        show_error(&cmd_options);
        return 1;
    }

    return process(&cmd_options);
}
```

A.1.2. constants.h

```
#ifndef __TP0_CONSTANTS_H__
#define __TP0_CONSTANTS_H__

// BOOLEAN VALUES
#define TRUE 1
#define FALSE 0

// RESULT VALUES
#define OK 0
#define ERROR 1

// PROCESS MODES
#define ENCODE 1
#define DECODE 0

// ERROR CODES
#define INVALID_ARGUMENT -1
#define NO_ARGUMENTS -2
#define EMPTY_INPUT -3
#define INVALID_FILE_LENGTH -4
#define INVALID_CHARS -5


#endif
```

A.1.3. command.h

```
#ifndef __COMMAND_H__
#define __COMMAND_H__

#include "file.h"
#include "constants.h"

typedef struct {
    File input_file;
    File output_file;
    const char *input_path;
    const char *output_path;
    char encode_option;
    char error_condition;
} command_options_st;

void command_create(command_options_st *opt);

void set_input_file(command_options_st *opt, const char *input);

void set_output_file(command_options_st *opt, const char *output);

void set_encode(command_options_st *opt);
```

```
23
24 void set_decode(command_options_st *opt);
25
26 void set_error(command_options_st *opt, char error_condition);
27
28 int has_errors(command_options_st *opt);
29
30 void show_error(command_options_st *opt);
31
32 void show_help();
33
34 void show_version();
35
36 char process(command_options_st *opt);
37
38 char _do_encode_decode(command_options_st *opt);
39
40 #endif
```

A.1.4. command.c

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdbool.h>
4  #include "command.h"
5  #include "encode.h"
6
7  void command_create(command_options_st *opt) {
8      create_file(&opt->input_file);
9      create_file(&opt->output_file);
10     opt->error_condition = OK;
11     opt->encode_option = ENCODE;
12     opt->input_path = 0;
13     opt->output_path = 0;
14 }
15
16 void set_input_file(command_options_st *opt, const char *input) {
17     opt->input_path = input;
18 }
19
20 void set_output_file(command_options_st *opt, const char *output) {
21     opt->output_path = output;
22 }
23
24 void set_encode(command_options_st *opt) {
25     opt->encode_option = ENCODE;
26 }
27
28 void set_decode(command_options_st *opt) {
29     opt->encode_option = DECODE;
30 }
31
32 void set_error(command_options_st *opt, char error_condition) {
33     opt->error_condition = error_condition;
34 }
35
36 int has_errors(command_options_st *opt) {
37     if (opt->error_condition != OK) {
38         return ERROR;
39     }
40     return OK;
41 }
42
43 void show_error(command_options_st *opt) {
44     char *error_message = NULL;
45     bool should_show_help = false;
46     if (opt->error_condition == INVALID_ARGUMENT) {
47         error_message = "Argumentos Invalidos!\n\n";
```

```
48          should_show_help = true;
49      } else if (opt->error_condition == NO_ARGUMENTS) {
50          error_message = "No se recibieron Argumentos!\n\n";
51          should_show_help = true;
52      } else if (opt->error_condition == INVALID_FILE_LENGTH) {
53          error_message = "Longitud de Archivo de Entrada Invalido!\n\n";
54      } else if (opt->error_condition == INVALID_CHARS) {
55          error_message = "Caracteres invalidos en Archivo Codificado!\n\n";
56      }
57
58      fprintf(stderr, "%s", error_message);
59      if (should_show_help) {
60          show_help();
61      }
62  }
63
64  void show_help() {
65      printf("Options:\n");
66      printf("\t-V,\t--version\tPrint version and quit.\n");
67      printf("\t-h,\t--help\t\tPrint this information.\n");
68      printf("\t-i,\t--input\t\tLocation of the input file.\n");
69      printf("\t-o,\t--output\tLocation of the output file.\n");
70      printf("\t-d,\t--decode\tDecode a base64-encoded file (default is encode).\n");
71      printf("Examples:\n");
72      printf("\ttp0 -i ~/input -o ~/output\n");
73      printf("\ttp0 --decode\n");
74  }
75
76  void show_version() {
77      printf("Version: 1.0\n");
78  }
79
80  char process(command_options_st *opt) {
81      if (open_file_read(&opt->input_file, opt->input_path) == ERROR) {
82          return ERROR;
83      }
84      if (open_file_write(&opt->output_file, opt->output_path) == ERROR) {
85          close_file(&opt->input_file);
86          return ERROR;
87      }
88
89      char result = _do_encode_decode(opt);
90      close_file(&opt->input_file);
91      close_file(&opt->output_file);
92      return result;
93  }
94
95  char _do_encode_decode(command_options_st *opt) {
96      unsigned char buf_decoded[3];
97      unsigned char buf_encoded[4];
98      unsigned char count = 0;
99
100     if (opt->encode_option == ENCODE) {
101         while (!file_eof(&opt->input_file)) {
102             memset(buf_decoded, 0, 3);
103             unsigned int read = file_read(&opt->input_file, buf_decoded, 3);
104             if (read > 0) {
105                 Encode(buf_decoded, read, buf_encoded);
106                 file_write(&opt->output_file, buf_encoded, 4);
107                 ++count;
108                 if (count == 18) // 19 * 4 =  76 bytes
109                 {
110                     file_write(&opt->output_file, (unsigned char *) "\n", 1);
111                     count = 0;
112                 }
113             }
114         }
```

```
115     }
116
117     if (opt->encode_option == DECODE) {
118         while (!file_eof(&opt->input_file) && !has_errors(opt)) {
119             unsigned int read = file_read(&opt->input_file, buf_encoded, 4);
120             if (read > 0) {
121                 if (read != 4) {
122                     set_error(opt, INVALID_FILE_LENGTH);
123                     show_error(opt);
124                 } else {
125                     ++count;
126                     if (count == 18) { // 19 * 4 =  76 bytes
127                         unsigned char aux;
128                         file_read(&opt->input_file, &aux, 1);
129                         count = 0;
130                     }
131
132                     if (Decode(buf_encoded, buf_decoded)) {
133                         char aux = 0;
134                         if (buf_encoded[2] == '=') {
135                             ++aux;
136                         }
137                         if (buf_encoded[3] == '=') {
138                             ++aux;
139                         }
140
141                         file_write(&opt->output_file, buf_decoded, 3 - aux);
142                     } else {
143                         set_error(opt, INVALID_CHARS);
144                         show_error(opt);
145                         unsigned int i;
146                         for (i = 0; i < 4; ++i) {
147                             fprintf(stderr, "%c", buf_encoded[i]);
148                         }
149                     }
150                 }
151
152             }
153         }
154     }
155
156     return opt->error_condition;
157 }
```

A.1.5. file.h

```
1  #ifndef __FILE_H__
2  #define __FILE_H__
3
4  #include <stdio.h>
5  #include "constants.h"
6
7
8  typedef struct {
9      FILE *file;
10     char eof;
11 } File;
12
13
14 void create_file(File *file);
15
16
17 char open_file_read(File *file, const char *route);
18
19
20 char open_file_write(File *file, const char *route);
21
22
```

```
int close_file(File *file);


unsigned int file_read(File *file, unsigned char *buffer, unsigned int length);


void file_write(File *file, unsigned char *buffer, unsigned int length);


int file_eof(File *file);

#endif
```

A.1.6. file.c

```
#include "file.h"
#include <stdlib.h>
#include <string.h>
#include <errno.h>

void create_file(File *file) {
    file->file = 0;
    file->eof = 0;
}

char open_file_read(File *file, const char *route) {
    if (route == NULL) {
        file->file = stdin;
    } else {
        file->file = fopen(route, "rb");
        if (file->file == NULL) {
            int err = errno;
            fprintf(stderr, "Error al abrir archivo: %s\n", strerror(err));
            return ERROR;
        }
    }
    return OK;
}


char open_file_write(File *file, const char *route) {
    if (route == NULL) {
        file->file = stdout;
    } else {
        file->file = fopen(route, "wb");
        if (file->file == NULL) {
            int err = errno;
            fprintf(stderr, "Error al abrir archivo: %s\n", strerror(err));
            return ERROR;
        }
    }
    return OK;
}


int close_file(File *file) {
    if (file->file == stdin || file->file == stdout) return OK;

    int result = fclose(file->file);
    if (result == EOF) {
        int err = errno;
        fprintf(stderr, "Error al cerrar archivo: %s\n", strerror(err));
        return ERROR;
    }
    return OK;
}


```

```
54 unsigned int file_read(File *file, unsigned char *buffer, unsigned int length) {
55     unsigned int result = 0;
56     if (!file_eof(file)) {
57         result = (unsigned int) fread(buffer, sizeof(char), length, file->file);
58         if (feof(file->file)) {
59             file->eof = 1;
60         }
61     }
62
63     return result;
64 }
65
66
67 void file_write(File *file, unsigned char *buffer, unsigned int length) {
68     fwrite(buffer, sizeof(char), length, file->file);
69 }
70
71
72 int file_eof(File *file) {
73     return file->eof;
74 }
```

A.1.7. encode.h

```
1  #ifndef __ENCODE_H__
2  #define __ENCODE_H__
3
4  /**
5   * Recibe 3 caracteres en buffer y los convierte en 4 caracteres codificados en output.
6   * Pre: el buffer contiene length caracteres (1 a 3) y todos los caracteres son validos
7   * Post: retorna un buffer de 4 byte con los caracteres en base64.
8   */
9  void Encode(const unsigned char *buffer, unsigned int length, unsigned char *output);
10
11 /**
12  * Devuelve un buffer de 3 caracteres recibiendo los 4 caracteres codificados en input.
13  * Pre:
14  * Post:
15  */
16 unsigned char Decode(unsigned char *buf_input, unsigned char *buf_output);
17
18 #endif // __ENCODE_H__
```

A.1.8. encode.c

```
1  #define BASE64_END '='
2  #define DECODE_ERROR 100
3
4  static unsigned char encoding_table[] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
5                                           'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
6                                           'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
7                                           'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
8                                           'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
9                                           'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
10                                          'w', 'x', 'y', 'z', '0', '1', '2', '3',
11                                          '4', '5', '6', '7', '8', '9', '+', '/'};
12
13 static int encoding_table_size = 64;
14
15
16 void Encode(const unsigned char *buffer, unsigned int length, unsigned char *output) {
17     unsigned char b1 = buffer[0];
18     unsigned char b2 = buffer[1];
19     unsigned char b3 = buffer[2];
20     //I retrieve the first 6 bits and operate..
21     unsigned char b1aux = b1 >> 2;
22     //Recovered the first 6 bits, I look into the encoding in the table.
```

```
23      output[0] = encoding_table[(int) b1aux];
24      //I retrieve the next 6 bits.
25      unsigned char b2aux = b1 << 6;
26      b2aux = b2aux >> 2;
27      b2aux = b2aux | (b2 >> 4);
28      //I take look into the encoding table
29      output[1] = encoding_table[(int) b2aux];
30      output[2] = BASE64_END;
31      output[3] = BASE64_END;
32      if (length == 3) {
33          /*
34       * If I have 3 characters in the buffer I operate
35       * with the last 2 characters.
36       */
37          unsigned char b3aux = b3 >> 6;
38          unsigned char b3aux2 = b2 << 4;
39          b3aux2 = b3aux2 >> 2;
40          b3aux = b3aux | b3aux2;
41          //I take look into the encoding table.
42          output[2] = encoding_table[(int) b3aux];
43          unsigned char b4aux = b3 << 2;
44          b4aux = b4aux >> 2;
45          //I take look into the encoding table.
46          output[3] = encoding_table[(int) b4aux];
47      } else {
48          if (length == 2) {
49              /*
50               * In case of having only 2 characters in the buffer
51               * I recover the remaining character and place the end of the line(=)
52               */
53              unsigned char b3aux = b3 >> 6;
54              unsigned char b3aux2 = b2 << 4;
55              b3aux2 = b3aux2 >> 2;
56              b3aux = b3aux | b3aux2;
57              output[2] = encoding_table[(int) b3aux];
58          }
59      }
60  }
61
62  /**
63   * Returns the representation of the char in the table.
64   * pre: character is valid (belongs to table)
65   * is the character '='.
66   * post: returns the representation (int) of the character
67   * in the encoding table.
68   *
69   */
70  unsigned char DecodeChar(char character) {
71      unsigned char i;
72      for (i = 0; i < encoding_table_size; i++) {
73          if (encoding_table[i] == character) {
74              return i;
75          }
76      }
77      if (character == '=') {
78          return 0;
79      }
80      return DECODE_ERROR;
81  }
82
83  /**
84   * Returns a buffer with size 3 with the 4 character base64 decode.
85   * Pre: the input buffer contains 4 characters. The output buffer has at least 3 characters
86   * Post: returns a buffer with size 3 ASCII characters. returns 0 if error 1 if ok
87   */
88  unsigned char Decode(unsigned char *buf_input, unsigned char *buf_output) {
89      unsigned char chars[4];
```

```
90      unsigned int i;
91      for (i = 0; i < 4; ++i) {
92          chars[i] = DecodeChar(buf_input[i]);
93          if (chars[i] == DECODE_ERROR)
94              return 0;
95      }
96
97      unsigned char char1_aux = chars[0] << 2;
98      //Take the last 2 bits of char2
99      unsigned char char2_aux = chars[1] >> 4;
100     char1_aux = char1_aux | char2_aux;
101     buf_output[0] = char1_aux;
102
103     //Take the last 4b of char2 and the first 4b of char3
104     char1_aux = chars[1] << 4;
105     char2_aux = chars[2] >> 2;
106     char2_aux = char1_aux | char2_aux;
107     buf_output[1] = char2_aux;
108
109     //Take the last 2b of char3 + the bits of char4
110     char1_aux = chars[2] << 6;
111     buf_output[2] = char1_aux | chars[3];
112     return 1;
113 }
```

A.2. Código MIPS32

A.2.1. tp0.S

```
1
2  tp0:     file format elf32-tradbigmips
3
4
5  Disassembly of section .init:
6
7  00000b28 <_init>:
8   b28:   3c1c0002        lui     gp,0x2
9   b2c:   279c9d48        addiu   gp,gp,-25272
10  b30:   0399e021        addu    gp,gp,t9
11  b34:   27bdffe0        addiu   sp,sp,-32
12  b38:   afbc0010        sw      gp,16(sp)
13  b3c:   afbf001c        sw      ra,28(sp)
14  b40:   8f8280d8        lw      v0,-32552(gp)
15  b44:   10400004        beqz    v0,b58 <_init+0x30>
16  b48:   00000000        nop
17  b4c:   8f9980d8        lw      t9,-32552(gp)
18  b50:   0320f809        jalr    t9
19  b54:   00000000        nop
20  b58:   8fbf001c        lw      ra,28(sp)
21  b5c:   03e00008        jr      ra
22  b60:   27bd0020        addiu   sp,sp,32
23
24 Disassembly of section .text:
25
26 00000b70 <__start>:
27      b70:       03e00025        move    zero,ra
28      b74:       04110001        bal     b7c <__start+0xc>
29      b78:       00000000        nop
30      b7c:       3c1c0002        lui     gp,0x2
31      b80:       279c9cf4        addiu   gp,gp,-25356
32      b84:       039fe021        addu    gp,gp,ra
33      b88:       0000f825        move    ra,zero
34      b8c:       8f848018        lw      a0,-32744(gp)
35      b90:       8fa50000        lw      a1,0(sp)
36      b94:       27a60004        addiu   a2,sp,4
37      b98:       2401fff8        li      at,-8
38      b9c:       03a1e824        and     sp,sp,at
39      ba0:       27bdffe0        addiu   sp,sp,-32
40      ba4:       8f87801c        lw      a3,-32740(gp)
41      ba8:       8f888020        lw      t0,-32736(gp)
42      bac:       afa80010        sw      t0,16(sp)
43      bb0:       afa20014        sw      v0,20(sp)
44      bb4:       afbd0018        sw      sp,24(sp)
45      bb8:       8f9980cc        lw      t9,-32564(gp)
46      bbc:       0320f809        jalr    t9
47      bc0:       00000000        nop
48
49 00000bc4 <hlt>:
50      bc4:       1000ffff        b       bc4 <hlt>
51      bc8:       00000000        nop
52      bcc:       00000000        nop
53
54 00000bd0 <deregister_tm_clones>:
55      bd0:       3c1c0002        lui     gp,0x2
56      bd4:       279c9ca0        addiu   gp,gp,-25440
57      bd8:       0399e021        addu    gp,gp,t9
58      bdc:       8f848028        lw      a0,-32728(gp)
59      be0:       8f828024        lw      v0,-32732(gp)
60      be4:       24842874        addiu   a0,a0,10356
61      be8:       24420003        addiu   v0,v0,3
62      bec:       00441023        subu    v0,v0,a0
63      bf0:       2c420007        sltiu   v0,v0,7
64      bf4:       14400005        bnez    v0,c0c <deregister_tm_clones+0x3c>
```

```
65      bf8:       8f9980f8        lw      t9,-32520(gp)
66      bfc:       13200003        beqz    t9,c0c <deregister_tm_clones+0x3c>
67      c00:       00000000        nop
68      c04:       03200008        jr      t9
69      c08:       00000000        nop
70      c0c:       03e00008        jr      ra
71      c10:       00000000        nop
72
73  00000c14 <register_tm_clones>:
74      c14:       3c1c0002        lui     gp,0x2
75      c18:       279c9c5c        addiu   gp,gp,-25508
76      c1c:       0399e021        addu    gp,gp,t9
77      c20:       8f848028        lw      a0,-32728(gp)
78      c24:       8f858024        lw      a1,-32732(gp)
79      c28:       24842874        addiu   a0,a0,10356
80      c2c:       00a42823        subu    a1,a1,a0
81      c30:       00052883        sra     a1,a1,0x2
82      c34:       000517c2        srl     v0,a1,0x1f
83      c38:       00452821        addu    a1,v0,a1
84      c3c:       00052843        sra     a1,a1,0x1
85      c40:       10a00005        beqz    a1,c58 <register_tm_clones+0x44>
86      c44:       8f9980a8        lw      t9,-32600(gp)
87      c48:       13200003        beqz    t9,c58 <register_tm_clones+0x44>
88      c4c:       00000000        nop
89      c50:       03200008        jr      t9
90      c54:       00000000        nop
91      c58:       03e00008        jr      ra
92      c5c:       00000000        nop
93
94  00000c60 <__do_global_dtors_aux>:
95      c60:       3c1c0002        lui     gp,0x2
96      c64:       279c9c10        addiu   gp,gp,-25584
97      c68:       0399e021        addu    gp,gp,t9
98      c6c:       27bdffe0        addiu   sp,sp,-32
99      c70:       afb00018        sw      s0,24(sp)
100     c74:       8f908028        lw      s0,-32728(gp)
101     c78:       afbc0010        sw      gp,16(sp)
102     c7c:       afbf001c        sw      ra,28(sp)
103     c80:       92022980        lbu     v0,10624(s0)
104     c84:       1440000d        bnez    v0,cbc <__do_global_dtors_aux+0x5c>
105     c88:       8f8280fc        lw      v0,-32516(gp)
106     c8c:       10400005        beqz    v0,ca4 <__do_global_dtors_aux+0x44>
107     c90:       8f82802c        lw      v0,-32724(gp)
108     c94:       8f9980fc        lw      t9,-32516(gp)
109     c98:       0320f809        jalr    t9
110     c9c:       8c440000        lw      a0,0(v0)
111     ca0:       8fbc0010        lw      gp,16(sp)
112     ca4:       8f998030        lw      t9,-32720(gp)
113     ca8:       27390bd0        addiu   t9,t9,3024
114     cac:       0411ffc8        bal     bd0 <deregister_tm_clones>
115     cb0:       00000000        nop
116     cb4:       24020001        li      v0,1
117     cb8:       a2022980        sb      v0,10624(s0)
118     cbc:       8fbf001c        lw      ra,28(sp)
119     cc0:       8fb00018        lw      s0,24(sp)
120     cc4:       03e00008        jr      ra
121     cc8:       27bd0020        addiu   sp,sp,32
122
123 00000ccc <frame_dummy>:
124     ccc:       3c1c0002        lui     gp,0x2
125     cd0:       279c9ba4        addiu   gp,gp,-25692
126     cd4:       0399e021        addu    gp,gp,t9
127     cd8:       8f828028        lw      v0,-32728(gp)
128     cdc:       27bdffe0        addiu   sp,sp,-32
129     ce0:       244427bc        addiu   a0,v0,10172
130     ce4:       afbc0010        sw      gp,16(sp)
131     ce8:       afbf001c        sw      ra,28(sp)
```

```
132      cec:       8c820000        lw      v0,0(a0)
133      cf0:       14400006        bnez    v0,d0c <frame_dummy+0x40>
134      cf4:       8f9980b0        lw      t9,-32592(gp)
135      cf8:       8f998030        lw      t9,-32720(gp)
136      cfc:       8fbf001c        lw      ra,28(sp)
137      d00:       27390c14        addiu   t9,t9,3092
138      d04:       1000ffc3        b       c14 <register_tm_clones>
139      d08:       27bd0020        addiu   sp,sp,32
140      d0c:       1320fffa        beqz    t9,cf8 <frame_dummy+0x2c>
141      d10:       00000000        nop
142      d14:       0320f809        jalr    t9
143      d18:       00000000        nop
144      d1c:       1000fff6        b       cf8 <frame_dummy+0x2c>
145      d20:       8fbc0010        lw      gp,16(sp)
146         ...
147
148 00000d30 <command_create>:
149 #include <string.h>
150 #include <stdbool.h>
151 #include "command.h"
152 #include "encode.h"
153
154 void command_create(command_options_st *opt) {
155      d30:       3c1c0002        lui     gp,0x2
156      d34:       279c9b40        addiu   gp,gp,-25792
157      d38:       0399e021        addu    gp,gp,t9
158      d3c:       27bdffe0        addiu   sp,sp,-32
159      d40:       afbf001c        sw      ra,28(sp)
160      d44:       afbe0018        sw      s8,24(sp)
161      d48:       03a0f025        move    s8,sp
162      d4c:       afbc0010        sw      gp,16(sp)
163      d50:       afc40020        sw      a0,32(s8)
164     create_file(&opt->input_file);
165      d54:       8fc20020        lw      v0,32(s8)
166      d58:       00402025        move    a0,v0
167      d5c:       8f828034        lw      v0,-32716(gp)
168      d60:       0040c825        move    t9,v0
169      d64:       04110376        bal     1b40 <create_file>
170      d68:       00000000        nop
171      d6c:       8fdc0010        lw      gp,16(s8)
172     create_file(&opt->output_file);
173      d70:       8fc20020        lw      v0,32(s8)
174      d74:       24420008        addiu   v0,v0,8
175      d78:       00402025        move    a0,v0
176      d7c:       8f828034        lw      v0,-32716(gp)
177      d80:       0040c825        move    t9,v0
178      d84:       0411036e        bal     1b40 <create_file>
179      d88:       00000000        nop
180      d8c:       8fdc0010        lw      gp,16(s8)
181     opt->error_condition = OK;
182      d90:       8fc20020        lw      v0,32(s8)
183      d94:       a0400019        sb      zero,25(v0)
184     opt->encode_option = ENCODE;
185      d98:       8fc20020        lw      v0,32(s8)
186      d9c:       24030001        li      v1,1
187      da0:       a0430018        sb      v1,24(v0)
188     opt->input_path = 0;
189      da4:       8fc20020        lw      v0,32(s8)
190      da8:       ac400010        sw      zero,16(v0)
191     opt->output_path = 0;
192      dac:       8fc20020        lw      v0,32(s8)
193      db0:       ac400014        sw      zero,20(v0)
194 }
195      db4:       00000000        nop
196      db8:       03c0e825        move    sp,s8
197      dbc:       8fbf001c        lw      ra,28(sp)
198      dc0:       8fbe0018        lw      s8,24(sp)
```

```
199       dc4:       27bd0020        addiu   sp,sp,32
200       dc8:       03e00008        jr      ra
201       dcc:       00000000        nop
202
203 00000dd0 <set_input_file>:
204
205 void set_input_file(command_options_st *opt, const char *input) {
206       dd0:       27bdfff8        addiu   sp,sp,-8
207       dd4:       afbe0004        sw      s8,4(sp)
208       dd8:       03a0f025        move    s8,sp
209       ddc:       afc40008        sw      a0,8(s8)
210       de0:       afc5000c        sw      a1,12(s8)
211     opt->input_path = input;
212       de4:       8fc20008        lw      v0,8(s8)
213       de8:       8fc3000c        lw      v1,12(s8)
214       dec:       ac430010        sw      v1,16(v0)
215 }
216       df0:       00000000        nop
217       df4:       03c0e825        move    sp,s8
218       df8:       8fbe0004        lw      s8,4(sp)
219       dfc:       27bd0008        addiu   sp,sp,8
220       e00:       03e00008        jr      ra
221       e04:       00000000        nop
222
223 00000e08 <set_output_file>:
224
225 void set_output_file(command_options_st *opt, const char *output) {
226       e08:       27bdfff8        addiu   sp,sp,-8
227       e0c:       afbe0004        sw      s8,4(sp)
228       e10:       03a0f025        move    s8,sp
229       e14:       afc40008        sw      a0,8(s8)
230       e18:       afc5000c        sw      a1,12(s8)
231     opt->output_path = output;
232       e1c:       8fc20008        lw      v0,8(s8)
233       e20:       8fc3000c        lw      v1,12(s8)
234       e24:       ac430014        sw      v1,20(v0)
235 }
236       e28:       00000000        nop
237       e2c:       03c0e825        move    sp,s8
238       e30:       8fbe0004        lw      s8,4(sp)
239       e34:       27bd0008        addiu   sp,sp,8
240       e38:       03e00008        jr      ra
241       e3c:       00000000        nop
242
243 00000e40 <set_encode>:
244
245 void set_encode(command_options_st *opt) {
246       e40:       27bdfff8        addiu   sp,sp,-8
247       e44:       afbe0004        sw      s8,4(sp)
248       e48:       03a0f025        move    s8,sp
249       e4c:       afc40008        sw      a0,8(s8)
250     opt->encode_option = ENCODE;
251       e50:       8fc20008        lw      v0,8(s8)
252       e54:       24030001        li      v1,1
253       e58:       a0430018        sb      v1,24(v0)
254 }
255       e5c:       00000000        nop
256       e60:       03c0e825        move    sp,s8
257       e64:       8fbe0004        lw      s8,4(sp)
258       e68:       27bd0008        addiu   sp,sp,8
259       e6c:       03e00008        jr      ra
260       e70:       00000000        nop
261
262 00000e74 <set_decode>:
263
264 void set_decode(command_options_st *opt) {
265       e74:       27bdfff8        addiu   sp,sp,-8
```

```
266      e78:       afbe0004        sw      s8,4(sp)
267      e7c:       03a0f025        move    s8,sp
268      e80:       afc40008        sw      a0,8(s8)
269     opt->encode_option = DECODE;
270      e84:       8fc20008        lw      v0,8(s8)
271      e88:       a0400018        sb      zero,24(v0)
272  }
273      e8c:       00000000        nop
274      e90:       03c0e825        move    sp,s8
275      e94:       8fbe0004        lw      s8,4(sp)
276      e98:       27bd0008        addiu   sp,sp,8
277      e9c:       03e00008        jr      ra
278      ea0:       00000000        nop
279
280  00000ea4 <set_error>:
281
282  void set_error(command_options_st *opt, char error_condition) {
283      ea4:       27bdfff8        addiu   sp,sp,-8
284      ea8:       afbe0004        sw      s8,4(sp)
285      eac:       03a0f025        move    s8,sp
286      eb0:       afc40008        sw      a0,8(s8)
287      eb4:       00a01025        move    v0,a1
288      eb8:       a3c2000c        sb      v0,12(s8)
289     opt->error_condition = error_condition;
290      ebc:       8fc20008        lw      v0,8(s8)
291      ec0:       93c3000c        lbu     v1,12(s8)
292      ec4:       a0430019        sb      v1,25(v0)
293  }
294      ec8:       00000000        nop
295      ecc:       03c0e825        move    sp,s8
296      ed0:       8fbe0004        lw      s8,4(sp)
297      ed4:       27bd0008        addiu   sp,sp,8
298      ed8:       03e00008        jr      ra
299      edc:       00000000        nop
300
301  00000ee0 <has_errors>:
302
303  int has_errors(command_options_st *opt) {
304      ee0:       27bdfff8        addiu   sp,sp,-8
305      ee4:       afbe0004        sw      s8,4(sp)
306      ee8:       03a0f025        move    s8,sp
307      eec:       afc40008        sw      a0,8(s8)
308     if (opt->error_condition != OK) {
309      ef0:       8fc20008        lw      v0,8(s8)
310      ef4:       80420019        lb      v0,25(v0)
311      ef8:       10400004        beqz    v0,f0c <has_errors+0x2c>
312      efc:       00000000        nop
313         return ERROR;
314      f00:       24020001        li      v0,1
315      f04:       10000002        b       f10 <has_errors+0x30>
316      f08:       00000000        nop
317     }
318     return OK;
319      f0c:       00001025        move    v0,zero
320  }
321      f10:       03c0e825        move    sp,s8
322      f14:       8fbe0004        lw      s8,4(sp)
323      f18:       27bd0008        addiu   sp,sp,8
324      f1c:       03e00008        jr      ra
325      f20:       00000000        nop
326
327  00000f24 <show_error>:
328
329  void show_error(command_options_st *opt) {
330      f24:       3c1c0002        lui     gp,0x2
331      f28:       279c994c        addiu   gp,gp,-26292
332      f2c:       0399e021        addu    gp,gp,t9
```

```
f30:       27bdffd8        addiu   sp,sp,-40
f34:       afbf0024        sw      ra,36(sp)
f38:       afbe0020        sw      s8,32(sp)
f3c:       03a0f025        move    s8,sp
f40:       afbc0010        sw      gp,16(sp)
f44:       afc40028        sw      a0,40(s8)
    char *error_message = NULL;
f48:       afc00018        sw      zero,24(s8)
    bool should_show_help = false;
f4c:       a3c0001c        sb      zero,28(s8)
    if (opt->error_condition == INVALID_ARGUMENT) {
f50:       8fc20028        lw      v0,40(s8)
f54:       80430019        lb      v1,25(v0)
f58:       2402ffff        li      v0,-1
f5c:       14620008        bne     v1,v0,f80 <show_error+0x5c>
f60:       00000000        nop
        error_message = "Argumentos Invalidos!\n\n";
f64:       8f828030        lw      v0,-32720(gp)
f68:       244224e0        addiu   v0,v0,9440
f6c:       afc20018        sw      v0,24(s8)
        should_show_help = true;
f70:       24020001        li      v0,1
f74:       a3c2001c        sb      v0,28(s8)
f78:       1000001f        b       ff8 <show_error+0xd4>
f7c:       00000000        nop
    } else if (opt->error_condition == NO_ARGUMENTS) {
f80:       8fc20028        lw      v0,40(s8)
f84:       80430019        lb      v1,25(v0)
f88:       2402fffe        li      v0,-2
f8c:       14620008        bne     v1,v0,fb0 <show_error+0x8c>
f90:       00000000        nop
        error_message = "No se recibieron Argumentos!\n\n";
f94:       8f828030        lw      v0,-32720(gp)
f98:       244224f8        addiu   v0,v0,9464
f9c:       afc20018        sw      v0,24(s8)
        should_show_help = true;
fa0:       24020001        li      v0,1
fa4:       a3c2001c        sb      v0,28(s8)
fa8:       10000013        b       ff8 <show_error+0xd4>
fac:       00000000        nop
    } else if (opt->error_condition == INVALID_FILE_LENGTH) {
fb0:       8fc20028        lw      v0,40(s8)
fb4:       80430019        lb      v1,25(v0)
fb8:       2402fffc        li      v0,-4
fbc:       14620006        bne     v1,v0,fd8 <show_error+0xb4>
fc0:       00000000        nop
        error_message = "Longitud de Archivo de Entrada Invalido!\n\n";
fc4:       8f828030        lw      v0,-32720(gp)
fc8:       24422518        addiu   v0,v0,9496
fcc:       afc20018        sw      v0,24(s8)
fd0:       10000009        b       ff8 <show_error+0xd4>
fd4:       00000000        nop
    } else if (opt->error_condition == INVALID_CHARS) {
fd8:       8fc20028        lw      v0,40(s8)
fdc:       80430019        lb      v1,25(v0)
fe0:       2402fffb        li      v0,-5
fe4:       14620004        bne     v1,v0,ff8 <show_error+0xd4>
fe8:       00000000        nop
        error_message = "Caracteres invalidos en Archivo Codificado!\n\n";
fec:       8f828030        lw      v0,-32720(gp)
ff0:       24422544        addiu   v0,v0,9540
ff4:       afc20018        sw      v0,24(s8)
    }

    fprintf(stderr, "%s", error_message);
ff8:       8f8280f0        lw      v0,-32528(gp)
ffc:       8c420000        lw      v0,0(v0)
```

```
    1000:       00402825        move    a1,v0
    1004:       8fc40018        lw      a0,24(s8)
    1008:       8f8280a0        lw      v0,-32608(gp)
    100c:       0040c825        move    t9,v0
    1010:       0320f809        jalr    t9
    1014:       00000000        nop
    1018:       8fdc0010        lw      gp,16(s8)
    if (should_show_help) {
    101c:       93c2001c        lbu     v0,28(s8)
    1020:       10400006        beqz    v0,103c <show_error+0x118>
    1024:       00000000        nop
        show_help();
    1028:       8f828038        lw      v0,-32712(gp)
    102c:       0040c825        move    t9,v0
    1030:       04110009        bal     1058 <show_help>
    1034:       00000000        nop
    1038:       8fdc0010        lw      gp,16(s8)
    }
}
    103c:       00000000        nop
    1040:       03c0e825        move    sp,s8
    1044:       8fbf0024        lw      ra,36(sp)
    1048:       8fbe0020        lw      s8,32(sp)
    104c:       27bd0028        addiu   sp,sp,40
    1050:       03e00008        jr      ra
    1054:       00000000        nop

00001058 <show_help>:

void show_help() {
    1058:       3c1c0002        lui     gp,0x2
    105c:       279c9818        addiu   gp,gp,-26600
    1060:       0399e021        addu    gp,gp,t9
    1064:       27bdffe0        addiu   sp,sp,-32
    1068:       afbf001c        sw      ra,28(sp)
    106c:       afbe0018        sw      s8,24(sp)
    1070:       03a0f025        move    s8,sp
    1074:       afbc0010        sw      gp,16(sp)
    printf("Options:\n");
    1078:       8f828030        lw      v0,-32720(gp)
    107c:       24442574        addiu   a0,v0,9588
    1080:       8f8280e0        lw      v0,-32544(gp)
    1084:       0040c825        move    t9,v0
    1088:       0320f809        jalr    t9
    108c:       00000000        nop
    1090:       8fdc0010        lw      gp,16(s8)
    printf("\t-V,\t--version\tPrint version and quit.\n");
    1094:       8f828030        lw      v0,-32720(gp)
    1098:       24442580        addiu   a0,v0,9600
    109c:       8f8280e0        lw      v0,-32544(gp)
    10a0:       0040c825        move    t9,v0
    10a4:       0320f809        jalr    t9
    10a8:       00000000        nop
    10ac:       8fdc0010        lw      gp,16(s8)
    printf("\t-h,\t--help\t\tPrint this information.\n");
    10b0:       8f828030        lw      v0,-32720(gp)
    10b4:       244425a8        addiu   a0,v0,9640
    10b8:       8f8280e0        lw      v0,-32544(gp)
    10bc:       0040c825        move    t9,v0
    10c0:       0320f809        jalr    t9
    10c4:       00000000        nop
    10c8:       8fdc0010        lw      gp,16(s8)
    printf("\t-i,\t--input\t\tLocation of the input file.\n");
    10cc:       8f828030        lw      v0,-32720(gp)
    10d0:       244425d0        addiu   a0,v0,9680
    10d4:       8f8280e0        lw      v0,-32544(gp)
    10d8:       0040c825        move    t9,v0
```

```
467     10dc:       0320f809        jalr    t9
468     10e0:       00000000        nop
469     10e4:       8fdc0010        lw      gp,16(s8)
470     printf("\t-o,\t--output\tLocation of the output file.\n");
471     10e8:       8f828030        lw      v0,-32720(gp)
472     10ec:       244425fc        addiu   a0,v0,9724
473     10f0:       8f8280e0        lw      v0,-32544(gp)
474     10f4:       0040c825        move    t9,v0
475     10f8:       0320f809        jalr    t9
476     10fc:       00000000        nop
477     1100:       8fdc0010        lw      gp,16(s8)
478     printf("\t-d,\t--decode\tDecode a base64-encoded file (default is encode).\n");
479     1104:       8f828030        lw      v0,-32720(gp)
480     1108:       24442628        addiu   a0,v0,9768
481     110c:       8f8280e0        lw      v0,-32544(gp)
482     1110:       0040c825        move    t9,v0
483     1114:       0320f809        jalr    t9
484     1118:       00000000        nop
485     111c:       8fdc0010        lw      gp,16(s8)
486     printf("Examples:\n");
487     1120:       8f828030        lw      v0,-32720(gp)
488     1124:       24442668        addiu   a0,v0,9832
489     1128:       8f8280e0        lw      v0,-32544(gp)
490     112c:       0040c825        move    t9,v0
491     1130:       0320f809        jalr    t9
492     1134:       00000000        nop
493     1138:       8fdc0010        lw      gp,16(s8)
494     printf("\ttp0 -i ~/input -o ~/output\n");
495     113c:       8f828030        lw      v0,-32720(gp)
496     1140:       24442674        addiu   a0,v0,9844
497     1144:       8f8280e0        lw      v0,-32544(gp)
498     1148:       0040c825        move    t9,v0
499     114c:       0320f809        jalr    t9
500     1150:       00000000        nop
501     1154:       8fdc0010        lw      gp,16(s8)
502     printf("\ttp0 --decode\n");
503     1158:       8f828030        lw      v0,-32720(gp)
504     115c:       24442690        addiu   a0,v0,9872
505     1160:       8f8280e0        lw      v0,-32544(gp)
506     1164:       0040c825        move    t9,v0
507     1168:       0320f809        jalr    t9
508     116c:       00000000        nop
509     1170:       8fdc0010        lw      gp,16(s8)
510 }
511     1174:       00000000        nop
512     1178:       03c0e825        move    sp,s8
513     117c:       8fbf001c        lw      ra,28(sp)
514     1180:       8fbe0018        lw      s8,24(sp)
515     1184:       27bd0020        addiu   sp,sp,32
516     1188:       03e00008        jr      ra
517     118c:       00000000        nop
518
519 00001190 <show_version>:
520
521 void show_version() {
522     1190:       3c1c0002        lui     gp,0x2
523     1194:       279c96e0        addiu   gp,gp,-26912
524     1198:       0399e021        addu    gp,gp,t9
525     119c:       27bdffe0        addiu   sp,sp,-32
526     11a0:       afbf001c        sw      ra,28(sp)
527     11a4:       afbe0018        sw      s8,24(sp)
528     11a8:       03a0f025        move    s8,sp
529     11ac:       afbc0010        sw      gp,16(sp)
530     printf("Version: 1.0\n");
531     11b0:       8f828030        lw      v0,-32720(gp)
532     11b4:       244426a0        addiu   a0,v0,9888
533     11b8:       8f8280e0        lw      v0,-32544(gp)
```

```
534     11bc:       0040c825        move    t9,v0
535     11c0:       0320f809        jalr    t9
536     11c4:       00000000        nop
537     11c8:       8fdc0010        lw      gp,16(s8)
538 }
539     11cc:       00000000        nop
540     11d0:       03c0e825        move    sp,s8
541     11d4:       8fbf001c        lw      ra,28(sp)
542     11d8:       8fbe0018        lw      s8,24(sp)
543     11dc:       27bd0020        addiu   sp,sp,32
544     11e0:       03e00008        jr      ra
545     11e4:       00000000        nop
546
547 000011e8 <process>:
548
549 char process(command_options_st *opt) {
550     11e8:       3c1c0002        lui     gp,0x2
551     11ec:       279c9688        addiu   gp,gp,-27000
552     11f0:       0399e021        addu    gp,gp,t9
553     11f4:       27bdffd8        addiu   sp,sp,-40
554     11f8:       afbf0024        sw      ra,36(sp)
555     11fc:       afbe0020        sw      s8,32(sp)
556     1200:       03a0f025        move    s8,sp
557     1204:       afbc0010        sw      gp,16(sp)
558     1208:       afc40028        sw      a0,40(s8)
559     char result = open_file_read(&opt->input_file, opt->input_path);
560     120c:       8fc30028        lw      v1,40(s8)
561     1210:       8fc20028        lw      v0,40(s8)
562     1214:       8c420010        lw      v0,16(v0)
563     1218:       00402825        move    a1,v0
564     121c:       00602025        move    a0,v1
565     1220:       8f82803c        lw      v0,-32708(gp)
566     1224:       0040c825        move    t9,v0
567     1228:       04110253        bal     1b78 <open_file_read>
568     122c:       00000000        nop
569     1230:       8fdc0010        lw      gp,16(s8)
570     1234:       a3c20018        sb      v0,24(s8)
571     if (!result) {
572     1238:       83c20018        lb      v0,24(s8)
573     123c:       1440000d        bnez    v0,1274 <process+0x8c>
574     1240:       00000000        nop
575         result = open_file_write(&opt->output_file, opt->output_path);
576     1244:       8fc20028        lw      v0,40(s8)
577     1248:       24430008        addiu   v1,v0,8
578     124c:       8fc20028        lw      v0,40(s8)
579     1250:       8c420014        lw      v0,20(v0)
580     1254:       00402825        move    a1,v0
581     1258:       00602025        move    a0,v1
582     125c:       8f828040        lw      v0,-32704(gp)
583     1260:       0040c825        move    t9,v0
584     1264:       0411028a        bal     1c90 <open_file_write>
585     1268:       00000000        nop
586     126c:       8fdc0010        lw      gp,16(s8)
587     1270:       a3c20018        sb      v0,24(s8)
588     }
589
590     if (!result) {
591     1274:       83c20018        lb      v0,24(s8)
592     1278:       14400019        bnez    v0,12e0 <process+0xf8>
593     127c:       00000000        nop
594         result = _do_encode_decode(opt);
595     1280:       8fc40028        lw      a0,40(s8)
596     1284:       8f828044        lw      v0,-32700(gp)
597     1288:       0040c825        move    t9,v0
598     128c:       04110022        bal     1318 <_do_encode_decode>
599     1290:       00000000        nop
600     1294:       8fdc0010        lw      gp,16(s8)
```

```
    1298:       a3c20018        sb      v0,24(s8)
        close_file(&opt->input_file);
    129c:       8fc20028        lw      v0,40(s8)
    12a0:       00402025        move    a0,v0
    12a4:       8f828048        lw      v0,-32696(gp)
    12a8:       0040c825        move    t9,v0
    12ac:       041102be        bal     1da8 <close_file>
    12b0:       00000000        nop
    12b4:       8fdc0010        lw      gp,16(s8)
        close_file(&opt->output_file);
    12b8:       8fc20028        lw      v0,40(s8)
    12bc:       24420008        addiu   v0,v0,8
    12c0:       00402025        move    a0,v0
    12c4:       8f828048        lw      v0,-32696(gp)
    12c8:       0040c825        move    t9,v0
    12cc:       041102b6        bal     1da8 <close_file>
    12d0:       00000000        nop
    12d4:       8fdc0010        lw      gp,16(s8)
    12d8:       10000008        b       12fc <process+0x114>
    12dc:       00000000        nop
    } else {
        close_file(&opt->input_file);
    12e0:       8fc20028        lw      v0,40(s8)
    12e4:       00402025        move    a0,v0
    12e8:       8f828048        lw      v0,-32696(gp)
    12ec:       0040c825        move    t9,v0
    12f0:       041102ad        bal     1da8 <close_file>
    12f4:       00000000        nop
    12f8:       8fdc0010        lw      gp,16(s8)
    }

    return result;
    12fc:       83c20018        lb      v0,24(s8)
}
    1300:       03c0e825        move    sp,s8
    1304:       8fbf0024        lw      ra,36(sp)
    1308:       8fbe0020        lw      s8,32(sp)
    130c:       27bd0028        addiu   sp,sp,40
    1310:       03e00008        jr      ra
    1314:       00000000        nop

00001318 <_do_encode_decode>:

char _do_encode_decode(command_options_st *opt) {
    1318:       3c1c0002        lui     gp,0x2
    131c:       279c9558        addiu   gp,gp,-27304
    1320:       0399e021        addu    gp,gp,t9
    1324:       27bdffc0        addiu   sp,sp,-64
    1328:       afbf003c        sw      ra,60(sp)
    132c:       afbe0038        sw      s8,56(sp)
    1330:       03a0f025        move    s8,sp
    1334:       afbc0010        sw      gp,16(sp)
    1338:       afc40040        sw      a0,64(s8)
    unsigned char buf_decoded[3];
    unsigned char buf_encoded[4];
    unsigned char count = 0;
    133c:       a3c00018        sb      zero,24(s8)

    if (opt->encode_option == ENCODE) {
    1340:       8fc20040        lw      v0,64(s8)
    1344:       80430018        lb      v1,24(v0)
    1348:       24020001        li      v0,1
    134c:       1462004b        bne     v1,v0,147c <_do_encode_decode+0x164>
    1350:       00000000        nop
        while (!file_eof(&opt->input_file)) {
    1354:       10000040        b       1458 <_do_encode_decode+0x140>
    1358:       00000000        nop
```

```
668             memset(buf_decoded, 0, 3);
669     135c:   27c20028        addiu   v0,s8,40
670     1360:   24060003        li      a2,3
671     1364:   00002825        move    a1,zero
672     1368:   00402025        move    a0,v0
673     136c:   8f8280c0        lw      v0,-32576(gp)
674     1370:   0040c825        move    t9,v0
675     1374:   0320f809        jalr    t9
676     1378:   00000000        nop
677     137c:   8fdc0010        lw      gp,16(s8)
678             unsigned int read = file_read(&opt->input_file, buf_decoded, 3);
679     1380:   8fc20040        lw      v0,64(s8)
680     1384:   27c30028        addiu   v1,s8,40
681     1388:   24060003        li      a2,3
682     138c:   00602825        move    a1,v1
683     1390:   00402025        move    a0,v0
684     1394:   8f82804c        lw      v0,-32692(gp)
685     1398:   0040c825        move    t9,v0
686     139c:   041102cb        bal     1ecc <file_read>
687     13a0:   00000000        nop
688     13a4:   8fdc0010        lw      gp,16(s8)
689     13a8:   afc20020        sw      v0,32(s8)
690             if (read > 0) {
691     13ac:   8fc20020        lw      v0,32(s8)
692     13b0:   10400029        beqz    v0,1458 <_do_encode_decode+0x140>
693     13b4:   00000000        nop
694                 Encode(buf_decoded, read, buf_encoded);
695     13b8:   27c3002c        addiu   v1,s8,44
696     13bc:   27c20028        addiu   v0,s8,40
697     13c0:   00603025        move    a2,v1
698     13c4:   8fc50020        lw      a1,32(s8)
699     13c8:   00402025        move    a0,v0
700     13cc:   8f828050        lw      v0,-32688(gp)
701     13d0:   0040c825        move    t9,v0
702     13d4:   041100ce        bal     1710 <Encode>
703     13d8:   00000000        nop
704     13dc:   8fdc0010        lw      gp,16(s8)
705                 file_write(&opt->output_file, buf_encoded, 4);
706     13e0:   8fc20040        lw      v0,64(s8)
707     13e4:   24420008        addiu   v0,v0,8
708     13e8:   27c3002c        addiu   v1,s8,44
709     13ec:   24060004        li      a2,4
710     13f0:   00602825        move    a1,v1
711     13f4:   00402025        move    a0,v0
712     13f8:   8f828054        lw      v0,-32684(gp)
713     13fc:   0040c825        move    t9,v0
714     1400:   041102e6        bal     1f9c <file_write>
715     1404:   00000000        nop
716     1408:   8fdc0010        lw      gp,16(s8)
717                 ++count;
718     140c:   93c20018        lbu     v0,24(s8)
719     1410:   24420001        addiu   v0,v0,1
720     1414:   a3c20018        sb      v0,24(s8)
721                 if (count == 18) // 19 * 4 =  76 bytes
722     1418:   93c30018        lbu     v1,24(s8)
723     141c:   24020012        li      v0,18
724     1420:   1462000d        bne     v1,v0,1458 <_do_encode_decode+0x140>
725     1424:   00000000        nop
726                 {
727                     file_write(&opt->output_file, (unsigned char *) "\n", 1);
728     1428:   8fc20040        lw      v0,64(s8)
729     142c:   24430008        addiu   v1,v0,8
730     1430:   24060001        li      a2,1
731     1434:   8f828030        lw      v0,-32720(gp)
732     1438:   244526b0        addiu   a1,v0,9904
733     143c:   00602025        move    a0,v1
734     1440:   8f828054        lw      v0,-32684(gp)
```

```
735     1444:       0040c825        move    t9,v0
736     1448:       041102d4        bal     1f9c <file_write>
737     144c:       00000000        nop
738     1450:       8fdc0010        lw      gp,16(s8)
739                     count = 0;
740     1454:       a3c00018        sb      zero,24(s8)
741         while (!file_eof(&opt->input_file)) {
742     1458:       8fc20040        lw      v0,64(s8)
743     145c:       00402025        move    a0,v0
744     1460:       8f828058        lw      v0,-32680(gp)
745     1464:       0040c825        move    t9,v0
746     1468:       041102e9        bal     2010 <file_eof>
747     146c:       00000000        nop
748     1470:       8fdc0010        lw      gp,16(s8)
749     1474:       1040ffb9        beqz    v0,135c <_do_encode_decode+0x44>
750     1478:       00000000        nop
751                 }
752             }
753         }
754     }
755
756     if (opt->encode_option == DECODE) {
757     147c:       8fc20040        lw      v0,64(s8)
758     1480:       80420018        lb      v0,24(v0)
759     1484:       14400097        bnez    v0,16e4 <_do_encode_decode+0x3cc>
760     1488:       00000000        nop
761         while (!file_eof(&opt->input_file) && !has_errors(opt)) {
762     148c:       10000084        b       16a0 <_do_encode_decode+0x388>
763     1490:       00000000        nop
764             unsigned int read = file_read(&opt->input_file, buf_encoded, 4);
765     1494:       8fc20040        lw      v0,64(s8)
766     1498:       27c3002c        addiu   v1,s8,44
767     149c:       24060004        li      a2,4
768     14a0:       00602825        move    a1,v1
769     14a4:       00402025        move    a0,v0
770     14a8:       8f82804c        lw      v0,-32692(gp)
771     14ac:       0040c825        move    t9,v0
772     14b0:       04110286        bal     1ecc <file_read>
773     14b4:       00000000        nop
774     14b8:       8fdc0010        lw      gp,16(s8)
775     14bc:       afc20024        sw      v0,36(s8)
776             if (read > 0) {
777     14c0:       8fc20024        lw      v0,36(s8)
778     14c4:       10400076        beqz    v0,16a0 <_do_encode_decode+0x388>
779     14c8:       00000000        nop
780                 if (read != 4) {
781     14cc:       8fc30024        lw      v1,36(s8)
782     14d0:       24020004        li      v0,4
783     14d4:       10620010        beq     v1,v0,1518 <_do_encode_decode+0x200>
784     14d8:       00000000        nop
785                     set_error(opt, INVALID_FILE_LENGTH);
786     14dc:       2405fffc        li      a1,-4
787     14e0:       8fc40040        lw      a0,64(s8)
788     14e4:       8f82805c        lw      v0,-32676(gp)
789     14e8:       0040c825        move    t9,v0
790     14ec:       0411fe6d        bal     ea4 <set_error>
791     14f0:       00000000        nop
792     14f4:       8fdc0010        lw      gp,16(s8)
793                     show_error(opt);
794     14f8:       8fc40040        lw      a0,64(s8)
795     14fc:       8f828060        lw      v0,-32672(gp)
796     1500:       0040c825        move    t9,v0
797     1504:       0411fe87        bal     f24 <show_error>
798     1508:       00000000        nop
799     150c:       8fdc0010        lw      gp,16(s8)
800     1510:       10000063        b       16a0 <_do_encode_decode+0x388>
801     1514:       00000000        nop
```

```
802                 } else {
803                     ++count;
804     1518:       93c20018        lbu     v0,24(s8)
805     151c:       24420001        addiu   v0,v0,1
806     1520:       a3c20018        sb      v0,24(s8)
807                     if (count == 18) { // 19 * 4 =  76 bytes
808     1524:       93c30018        lbu     v1,24(s8)
809     1528:       24020012        li      v0,18
810     152c:       1462000c        bne     v1,v0,1560 <_do_encode_decode+0x248>
811     1530:       00000000        nop
812                         unsigned char aux;
813                         file_read(&opt->input_file, &aux, 1);
814     1534:       8fc20040        lw      v0,64(s8)
815     1538:       27c30030        addiu   v1,s8,48
816     153c:       24060001        li      a2,1
817     1540:       00602825        move    a1,v1
818     1544:       00402025        move    a0,v0
819     1548:       8f82804c        lw      v0,-32692(gp)
820     154c:       0040c825        move    t9,v0
821     1550:       0411025e        bal     1ecc <file_read>
822     1554:       00000000        nop
823     1558:       8fdc0010        lw      gp,16(s8)
824                         count = 0;
825     155c:       a3c00018        sb      zero,24(s8)
826                     }
827
828                     if (Decode(buf_encoded, buf_decoded)) {
829     1560:       27c30028        addiu   v1,s8,40
830     1564:       27c2002c        addiu   v0,s8,44
831     1568:       00602825        move    a1,v1
832     156c:       00402025        move    a0,v0
833     1570:       8f828064        lw      v0,-32668(gp)
834     1574:       0040c825        move    t9,v0
835     1578:       04110114        bal     19cc <Decode>
836     157c:       00000000        nop
837     1580:       8fdc0010        lw      gp,16(s8)
838     1584:       10400022        beqz    v0,1610 <_do_encode_decode+0x2f8>
839     1588:       00000000        nop
840                         char aux = 0;
841     158c:       a3c00019        sb      zero,25(s8)
842                         if (buf_encoded[2] == '=') {
843     1590:       93c3002e        lbu     v1,46(s8)
844     1594:       2402003d        li      v0,61
845     1598:       14620005        bne     v1,v0,15b0 <_do_encode_decode+0x298>
846     159c:       00000000        nop
847                             ++aux;
848     15a0:       93c20019        lbu     v0,25(s8)
849     15a4:       24420001        addiu   v0,v0,1
850     15a8:       304200ff        andi    v0,v0,0xff
851     15ac:       a3c20019        sb      v0,25(s8)
852                         }
853                         if (buf_encoded[3] == '=') {
854     15b0:       93c3002f        lbu     v1,47(s8)
855     15b4:       2402003d        li      v0,61
856     15b8:       14620005        bne     v1,v0,15d0 <_do_encode_decode+0x2b8>
857     15bc:       00000000        nop
858                             ++aux;
859     15c0:       93c20019        lbu     v0,25(s8)
860     15c4:       24420001        addiu   v0,v0,1
861     15c8:       304200ff        andi    v0,v0,0xff
862     15cc:       a3c20019        sb      v0,25(s8)
863                         }
864
865                         file_write(&opt->output_file, buf_decoded, 3 - aux);
866     15d0:       8fc20040        lw      v0,64(s8)
867     15d4:       24440008        addiu   a0,v0,8
868     15d8:       83c20019        lb      v0,25(s8)
```

```
869     15dc:       24030003        li      v1,3
870     15e0:       00621023        subu    v0,v1,v0
871     15e4:       00401825        move    v1,v0
872     15e8:       27c20028        addiu   v0,s8,40
873     15ec:       00603025        move    a2,v1
874     15f0:       00402825        move    a1,v0
875     15f4:       8f828054        lw      v0,-32684(gp)
876     15f8:       0040c825        move    t9,v0
877     15fc:       04110267        bal     1f9c <file_write>
878     1600:       00000000        nop
879     1604:       8fdc0010        lw      gp,16(s8)
880     1608:       10000025        b       16a0 <_do_encode_decode+0x388>
881     160c:       00000000        nop
882                     } else {
883                         set_error(opt, INVALID_CHARS);
884     1610:       2405fffb        li      a1,-5
885     1614:       8fc40040        lw      a0,64(s8)
886     1618:       8f82805c        lw      v0,-32676(gp)
887     161c:       0040c825        move    t9,v0
888     1620:       0411fe20        bal     ea4 <set_error>
889     1624:       00000000        nop
890     1628:       8fdc0010        lw      gp,16(s8)
891                     show_error(opt);
892     162c:       8fc40040        lw      a0,64(s8)
893     1630:       8f828060        lw      v0,-32672(gp)
894     1634:       0040c825        move    t9,v0
895     1638:       0411fe3a        bal     f24 <show_error>
896     163c:       00000000        nop
897     1640:       8fdc0010        lw      gp,16(s8)
898                     unsigned int i;
899                     for (i = 0; i < 4; ++i) {
900     1644:       afc0001c        sw      zero,28(s8)
901     1648:       10000011        b       1690 <_do_encode_decode+0x378>
902     164c:       00000000        nop
903                         fprintf(stderr, "%c", buf_encoded[i]);
904     1650:       8f8280f0        lw      v0,-32528(gp)
905     1654:       8c440000        lw      a0,0(v0)
906     1658:       8fc2001c        lw      v0,28(s8)
907     165c:       27c30018        addiu   v1,s8,24
908     1660:       00621021        addu    v0,v1,v0
909     1664:       90420014        lbu     v0,20(v0)
910     1668:       00802825        move    a1,a0
911     166c:       00402025        move    a0,v0
912     1670:       8f8280b4        lw      v0,-32588(gp)
913     1674:       0040c825        move    t9,v0
914     1678:       0320f809        jalr    t9
915     167c:       00000000        nop
916     1680:       8fdc0010        lw      gp,16(s8)
917                     for (i = 0; i < 4; ++i) {
918     1684:       8fc2001c        lw      v0,28(s8)
919     1688:       24420001        addiu   v0,v0,1
920     168c:       afc2001c        sw      v0,28(s8)
921     1690:       8fc2001c        lw      v0,28(s8)
922     1694:       2c420004        sltiu   v0,v0,4
923     1698:       1440ffed        bnez    v0,1650 <_do_encode_decode+0x338>
924     169c:       00000000        nop
925         while (!file_eof(&opt->input_file) && !has_errors(opt)) {
926     16a0:       8fc20040        lw      v0,64(s8)
927     16a4:       00402025        move    a0,v0
928     16a8:       8f828058        lw      v0,-32680(gp)
929     16ac:       0040c825        move    t9,v0
930     16b0:       04110257        bal     2010 <file_eof>
931     16b4:       00000000        nop
932     16b8:       8fdc0010        lw      gp,16(s8)
933     16bc:       14400009        bnez    v0,16e4 <_do_encode_decode+0x3cc>
934     16c0:       00000000        nop
935     16c4:       8fc40040        lw      a0,64(s8)
```

```
936     16c8:       8f828068        lw      v0,-32664(gp)
937     16cc:       0040c825        move    t9,v0
938     16d0:       0411fe03        bal     ee0 <has_errors>
939     16d4:       00000000        nop
940     16d8:       8fdc0010        lw      gp,16(s8)
941     16dc:       1040ff6d        beqz    v0,1494 <_do_encode_decode+0x17c>
942     16e0:       00000000        nop
943
944             }
945         }
946     }
947
948     return opt->error_condition;
949     16e4:       8fc20040        lw      v0,64(s8)
950     16e8:       80420019        lb      v0,25(v0)
951 }
952     16ec:       03c0e825        move    sp,s8
953     16f0:       8fbf003c        lw      ra,60(sp)
954     16f4:       8fbe0038        lw      s8,56(sp)
955     16f8:       27bd0040        addiu   sp,sp,64
956     16fc:       03e00008        jr      ra
957     1700:       00000000        nop
958         ...
959
960 00001710 <Encode>:
961                                          '4', '5', '6', '7', '8', '9', '+', '/'};
962
963 static int encoding_table_size = 64;
964
965
966 void Encode(const unsigned char *buffer, unsigned int length, unsigned char *output) {
967     1710:       3c1c0002        lui     gp,0x2
968     1714:       279c9160        addiu   gp,gp,-28320
969     1718:       0399e021        addu    gp,gp,t9
970     171c:       27bdffe0        addiu   sp,sp,-32
971     1720:       afbe001c        sw      s8,28(sp)
972     1724:       03a0f025        move    s8,sp
973     1728:       afbc0000        sw      gp,0(sp)
974     172c:       afc40020        sw      a0,32(s8)
975     1730:       afc50024        sw      a1,36(s8)
976     1734:       afc60028        sw      a2,40(s8)
977     unsigned char b1 = buffer[0];
978     1738:       8fc20020        lw      v0,32(s8)
979     173c:       90420000        lbu     v0,0(v0)
980     1740:       a3c20008        sb      v0,8(s8)
981     unsigned char b2 = buffer[1];
982     1744:       8fc20020        lw      v0,32(s8)
983     1748:       90420001        lbu     v0,1(v0)
984     174c:       a3c20009        sb      v0,9(s8)
985     unsigned char b3 = buffer[2];
986     1750:       8fc20020        lw      v0,32(s8)
987     1754:       90420002        lbu     v0,2(v0)
988     1758:       a3c2000a        sb      v0,10(s8)
989     //I retrieve the first 6 bits and operate..
990     unsigned char b1aux = b1 >> 2;
991     175c:       93c20008        lbu     v0,8(s8)
992     1760:       00021082        srl     v0,v0,0x2
993     1764:       a3c2000b        sb      v0,11(s8)
994     //Recovered the first 6 bits, I look into the encoding in the table.
995     output[0] = encoding_table[(int) b1aux];
996     1768:       93c3000b        lbu     v1,11(s8)
997     176c:       8f828028        lw      v0,-32728(gp)
998     1770:       244227d0        addiu   v0,v0,10192
999     1774:       00621021        addu    v0,v1,v0
1000    1778:       90430000        lbu     v1,0(v0)
1001    177c:       8fc20028        lw      v0,40(s8)
1002    1780:       a0430000        sb      v1,0(v0)
```

```
1003      //I retrieve the next 6 bits.
1004      unsigned char b2aux = b1 << 6;
1005      1784:       93c20008        lbu     v0,8(s8)
1006      1788:       00021180        sll     v0,v0,0x6
1007      178c:       a3c2000c        sb      v0,12(s8)
1008      b2aux = b2aux >> 2;
1009      1790:       93c2000c        lbu     v0,12(s8)
1010      1794:       00021082        srl     v0,v0,0x2
1011      1798:       a3c2000c        sb      v0,12(s8)
1012      b2aux = b2aux | (b2 >> 4);
1013      179c:       93c20009        lbu     v0,9(s8)
1014      17a0:       00021102        srl     v0,v0,0x4
1015      17a4:       304300ff        andi    v1,v0,0xff
1016      17a8:       93c2000c        lbu     v0,12(s8)
1017      17ac:       00621025        or      v0,v1,v0
1018      17b0:       a3c2000c        sb      v0,12(s8)
1019      //I take look into the encoding table
1020      output[1] = encoding_table[(int) b2aux];
1021      17b4:       8fc20028        lw      v0,40(s8)
1022      17b8:       24420001        addiu   v0,v0,1
1023      17bc:       93c4000c        lbu     a0,12(s8)
1024      17c0:       8f838028        lw      v1,-32728(gp)
1025      17c4:       246327d0        addiu   v1,v1,10192
1026      17c8:       00831821        addu    v1,a0,v1
1027      17cc:       90630000        lbu     v1,0(v1)
1028      17d0:       a0430000        sb      v1,0(v0)
1029      output[2] = BASE64_END;
1030      17d4:       8fc20028        lw      v0,40(s8)
1031      17d8:       24420002        addiu   v0,v0,2
1032      17dc:       2403003d        li      v1,61
1033      17e0:       a0430000        sb      v1,0(v0)
1034      output[3] = BASE64_END;
1035      17e4:       8fc20028        lw      v0,40(s8)
1036      17e8:       24420003        addiu   v0,v0,3
1037      17ec:       2403003d        li      v1,61
1038      17f0:       a0430000        sb      v1,0(v0)
1039      if (length == 3) {
1040      17f4:       8fc30024        lw      v1,36(s8)
1041      17f8:       24020003        li      v0,3
1042      17fc:       14620026        bne     v1,v0,1898 <Encode+0x188>
1043      1800:       00000000        nop
1044          /*
1045       * If I have 3 characters in the buffer I operate
1046       * with the last 2 characters.
1047       */
1048          unsigned char b3aux = b3 >> 6;
1049      1804:       93c2000a        lbu     v0,10(s8)
1050      1808:       00021182        srl     v0,v0,0x6
1051      180c:       a3c2000d        sb      v0,13(s8)
1052          unsigned char b3aux2 = b2 << 4;
1053      1810:       93c20009        lbu     v0,9(s8)
1054      1814:       00021100        sll     v0,v0,0x4
1055      1818:       a3c2000e        sb      v0,14(s8)
1056          b3aux2 = b3aux2 >> 2;
1057      181c:       93c2000e        lbu     v0,14(s8)
1058      1820:       00021082        srl     v0,v0,0x2
1059      1824:       a3c2000e        sb      v0,14(s8)
1060          b3aux = b3aux | b3aux2;
1061      1828:       93c3000d        lbu     v1,13(s8)
1062      182c:       93c2000e        lbu     v0,14(s8)
1063      1830:       00621025        or      v0,v1,v0
1064      1834:       a3c2000d        sb      v0,13(s8)
1065          //I take look into the encoding table.
1066          output[2] = encoding_table[(int) b3aux];
1067      1838:       8fc20028        lw      v0,40(s8)
1068      183c:       24420002        addiu   v0,v0,2
1069      1840:       93c4000d        lbu     a0,13(s8)
```

```
1070      1844:       8f838028        lw      v1,-32728(gp)
1071      1848:       246327d0        addiu   v1,v1,10192
1072      184c:       00831821        addu    v1,a0,v1
1073      1850:       90630000        lbu     v1,0(v1)
1074      1854:       a0430000        sb      v1,0(v0)
1075          unsigned char b4aux = b3 << 2;
1076      1858:       93c2000a        lbu     v0,10(s8)
1077      185c:       00021080        sll     v0,v0,0x2
1078      1860:       a3c2000f        sb      v0,15(s8)
1079          b4aux = b4aux >> 2;
1080      1864:       93c2000f        lbu     v0,15(s8)
1081      1868:       00021082        srl     v0,v0,0x2
1082      186c:       a3c2000f        sb      v0,15(s8)
1083          //I take look into the encoding table.
1084          output[3] = encoding_table[(int) b4aux];
1085      1870:       8fc20028        lw      v0,40(s8)
1086      1874:       24420003        addiu   v0,v0,3
1087      1878:       93c4000f        lbu     a0,15(s8)
1088      187c:       8f838028        lw      v1,-32728(gp)
1089      1880:       246327d0        addiu   v1,v1,10192
1090      1884:       00831821        addu    v1,a0,v1
1091      1888:       90630000        lbu     v1,0(v1)
1092      188c:       a0430000        sb      v1,0(v0)
1093              b3aux2 = b3aux2 >> 2;
1094              b3aux = b3aux | b3aux2;
1095              output[2] = encoding_table[(int) b3aux];
1096          }
1097      }
1098  }
1099      1890:       1000001a        b       18fc <Encode+0x1ec>
1100      1894:       00000000        nop
1101          if (length == 2) {
1102      1898:       8fc30024        lw      v1,36(s8)
1103      189c:       24020002        li      v0,2
1104      18a0:       14620016        bne     v1,v0,18fc <Encode+0x1ec>
1105      18a4:       00000000        nop
1106              unsigned char b3aux = b3 >> 6;
1107      18a8:       93c2000a        lbu     v0,10(s8)
1108      18ac:       00021182        srl     v0,v0,0x6
1109      18b0:       a3c20010        sb      v0,16(s8)
1110              unsigned char b3aux2 = b2 << 4;
1111      18b4:       93c20009        lbu     v0,9(s8)
1112      18b8:       00021100        sll     v0,v0,0x4
1113      18bc:       a3c20011        sb      v0,17(s8)
1114              b3aux2 = b3aux2 >> 2;
1115      18c0:       93c20011        lbu     v0,17(s8)
1116      18c4:       00021082        srl     v0,v0,0x2
1117      18c8:       a3c20011        sb      v0,17(s8)
1118              b3aux = b3aux | b3aux2;
1119      18cc:       93c30010        lbu     v1,16(s8)
1120      18d0:       93c20011        lbu     v0,17(s8)
1121      18d4:       00621025        or      v0,v1,v0
1122      18d8:       a3c20010        sb      v0,16(s8)
1123              output[2] = encoding_table[(int) b3aux];
1124      18dc:       8fc20028        lw      v0,40(s8)
1125      18e0:       24420002        addiu   v0,v0,2
1126      18e4:       93c40010        lbu     a0,16(s8)
1127      18e8:       8f838028        lw      v1,-32728(gp)
1128      18ec:       246327d0        addiu   v1,v1,10192
1129      18f0:       00831821        addu    v1,a0,v1
1130      18f4:       90630000        lbu     v1,0(v1)
1131      18f8:       a0430000        sb      v1,0(v0)
1132  }
1133      18fc:       00000000        nop
1134      1900:       03c0e825        move    sp,s8
1135      1904:       8fbe001c        lw      s8,28(sp)
1136      1908:       27bd0020        addiu   sp,sp,32
```

```
    190c:       03e00008        jr      ra
    1910:       00000000        nop

00001914 <DecodeChar>:
 * is the character '='.
 * post: returns the representation (int) of the character
 * in the encoding table.
 *
 */
unsigned char DecodeChar(char character) {
    1914:       3c1c0002        lui     gp,0x2
    1918:       279c8f5c        addiu   gp,gp,-28836
    191c:       0399e021        addu    gp,gp,t9
    1920:       27bdffe8        addiu   sp,sp,-24
    1924:       afbe0014        sw      s8,20(sp)
    1928:       03a0f025        move    s8,sp
    192c:       afbc0000        sw      gp,0(sp)
    1930:       00801025        move    v0,a0
    1934:       a3c20018        sb      v0,24(s8)
    unsigned char i;
    for (i = 0; i < encoding_table_size; i++) {
    1938:       a3c00008        sb      zero,8(s8)
    193c:       10000010        b       1980 <DecodeChar+0x6c>
    1940:       00000000        nop
        if (encoding_table[i] == character) {
    1944:       93c30008        lbu     v1,8(s8)
    1948:       8f828028        lw      v0,-32728(gp)
    194c:       244227d0        addiu   v0,v0,10192
    1950:       00621021        addu    v0,v1,v0
    1954:       90420000        lbu     v0,0(v0)
    1958:       00401825        move    v1,v0
    195c:       83c20018        lb      v0,24(s8)
    1960:       14620004        bne     v1,v0,1974 <DecodeChar+0x60>
    1964:       00000000        nop
            return i;
    1968:       93c20008        lbu     v0,8(s8)
    196c:       10000012        b       19b8 <DecodeChar+0xa4>
    1970:       00000000        nop
    for (i = 0; i < encoding_table_size; i++) {
    1974:       93c20008        lbu     v0,8(s8)
    1978:       24420001        addiu   v0,v0,1
    197c:       a3c20008        sb      v0,8(s8)
    1980:       93c30008        lbu     v1,8(s8)
    1984:       8f828028        lw      v0,-32728(gp)
    1988:       8c422810        lw      v0,10256(v0)
    198c:       0062102a        slt     v0,v1,v0
    1990:       1440ffec        bnez    v0,1944 <DecodeChar+0x30>
    1994:       00000000        nop
        }
    }
    if (character == '=') {
    1998:       83c30018        lb      v1,24(s8)
    199c:       2402003d        li      v0,61
    19a0:       14620004        bne     v1,v0,19b4 <DecodeChar+0xa0>
    19a4:       00000000        nop
        return 0;
    19a8:       00001025        move    v0,zero
    19ac:       10000002        b       19b8 <DecodeChar+0xa4>
    19b0:       00000000        nop
    }
    return DECODE_ERROR;
    19b4:       24020064        li      v0,100
}
    19b8:       03c0e825        move    sp,s8
    19bc:       8fbe0014        lw      s8,20(sp)
    19c0:       27bd0018        addiu   sp,sp,24
    19c4:       03e00008        jr      ra
```

```
    19c8:       00000000        nop

000019cc <Decode>:
/**
 * Returns a buffer with size 3 with the 4 character base64 decode.
 * Pre: the input buffer contains 4 characters. The output buffer has at least 3 characters
 * Post: returns a buffer with size 3 ASCII characters. returns 0 if error 1 if ok
 */
unsigned char Decode(unsigned char *buf_input, unsigned char *buf_output) {
    19cc:       3c1c0002        lui     gp,0x2
    19d0:       279c8ea4        addiu   gp,gp,-29020
    19d4:       0399e021        addu    gp,gp,t9
    19d8:       27bdffd0        addiu   sp,sp,-48
    19dc:       afbf002c        sw      ra,44(sp)
    19e0:       afbe0028        sw      s8,40(sp)
    19e4:       03a0f025        move    s8,sp
    19e8:       afbc0010        sw      gp,16(sp)
    19ec:       afc40030        sw      a0,48(s8)
    19f0:       afc50034        sw      a1,52(s8)
    unsigned char chars[4];
    unsigned int i;
    for (i = 0; i < 4; ++i) {
    19f4:       afc00018        sw      zero,24(s8)
    19f8:       1000001e        b       1a74 <Decode+0xa8>
    19fc:       00000000        nop
        chars[i] = DecodeChar(buf_input[i]);
    1a00:       8fc30030        lw      v1,48(s8)
    1a04:       8fc20018        lw      v0,24(s8)
    1a08:       00621021        addu    v0,v1,v0
    1a0c:       90420000        lbu     v0,0(v0)
    1a10:       7c021420        seb     v0,v0
    1a14:       00402025        move    a0,v0
    1a18:       8f82806c        lw      v0,-32660(gp)
    1a1c:       0040c825        move    t9,v0
    1a20:       0411ffbc        bal     1914 <DecodeChar>
    1a24:       00000000        nop
    1a28:       8fdc0010        lw      gp,16(s8)
    1a2c:       00402025        move    a0,v0
    1a30:       8fc20018        lw      v0,24(s8)
    1a34:       27c30018        addiu   v1,s8,24
    1a38:       00621021        addu    v0,v1,v0
    1a3c:       a0440008        sb      a0,8(v0)
        if (chars[i] == DECODE_ERROR)
    1a40:       8fc20018        lw      v0,24(s8)
    1a44:       27c30018        addiu   v1,s8,24
    1a48:       00621021        addu    v0,v1,v0
    1a4c:       90430008        lbu     v1,8(v0)
    1a50:       24020064        li      v0,100
    1a54:       14620004        bne     v1,v0,1a68 <Decode+0x9c>
    1a58:       00000000        nop
            return 0;
    1a5c:       00001025        move    v0,zero
    1a60:       1000002e        b       1b1c <Decode+0x150>
    1a64:       00000000        nop
    for (i = 0; i < 4; ++i) {
    1a68:       8fc20018        lw      v0,24(s8)
    1a6c:       24420001        addiu   v0,v0,1
    1a70:       afc20018        sw      v0,24(s8)
    1a74:       8fc20018        lw      v0,24(s8)
    1a78:       2c420004        sltiu   v0,v0,4
    1a7c:       1440ffe0        bnez    v0,1a00 <Decode+0x34>
    1a80:       00000000        nop
    }

    unsigned char char1_aux = chars[0] << 2;
    1a84:       93c20020        lbu     v0,32(s8)
    1a88:       00021080        sll     v0,v0,0x2
```

```
1271      1a8c:       a3c2001c        sb      v0,28(s8)
1272      //Take the last 2 bits of char2
1273      unsigned char char2_aux = chars[1] >> 4;
1274      1a90:       93c20021        lbu     v0,33(s8)
1275      1a94:       00021102        srl     v0,v0,0x4
1276      1a98:       a3c2001d        sb      v0,29(s8)
1277      char1_aux = char1_aux | char2_aux;
1278      1a9c:       93c3001c        lbu     v1,28(s8)
1279      1aa0:       93c2001d        lbu     v0,29(s8)
1280      1aa4:       00621025        or      v0,v1,v0
1281      1aa8:       a3c2001c        sb      v0,28(s8)
1282      buf_output[0] = char1_aux;
1283      1aac:       8fc20034        lw      v0,52(s8)
1284      1ab0:       93c3001c        lbu     v1,28(s8)
1285      1ab4:       a0430000        sb      v1,0(v0)
1286
1287      //Take the last 4b of char2 and the first 4b of char3
1288      char1_aux = chars[1] << 4;
1289      1ab8:       93c20021        lbu     v0,33(s8)
1290      1abc:       00021100        sll     v0,v0,0x4
1291      1ac0:       a3c2001c        sb      v0,28(s8)
1292      char2_aux = chars[2] >> 2;
1293      1ac4:       93c20022        lbu     v0,34(s8)
1294      1ac8:       00021082        srl     v0,v0,0x2
1295      1acc:       a3c2001d        sb      v0,29(s8)
1296      char2_aux = char1_aux | char2_aux;
1297      1ad0:       93c3001c        lbu     v1,28(s8)
1298      1ad4:       93c2001d        lbu     v0,29(s8)
1299      1ad8:       00621025        or      v0,v1,v0
1300      1adc:       a3c2001d        sb      v0,29(s8)
1301      buf_output[1] = char2_aux;
1302      1ae0:       8fc20034        lw      v0,52(s8)
1303      1ae4:       24420001        addiu   v0,v0,1
1304      1ae8:       93c3001d        lbu     v1,29(s8)
1305      1aec:       a0430000        sb      v1,0(v0)
1306
1307      //Take the last 2b of char3 + the bits of char4
1308      char1_aux = chars[2] << 6;
1309      1af0:       93c20022        lbu     v0,34(s8)
1310      1af4:       00021180        sll     v0,v0,0x6
1311      1af8:       a3c2001c        sb      v0,28(s8)
1312      buf_output[2] = char1_aux | chars[3];
1313      1afc:       8fc20034        lw      v0,52(s8)
1314      1b00:       24420002        addiu   v0,v0,2
1315      1b04:       93c40023        lbu     a0,35(s8)
1316      1b08:       93c3001c        lbu     v1,28(s8)
1317      1b0c:       00831825        or      v1,a0,v1
1318      1b10:       306300ff        andi    v1,v1,0xff
1319      1b14:       a0430000        sb      v1,0(v0)
1320      return 1;
1321      1b18:       24020001        li      v0,1
1322  }
1323      1b1c:       03c0e825        move    sp,s8
1324      1b20:       8fbf002c        lw      ra,44(sp)
1325      1b24:       8fbe0028        lw      s8,40(sp)
1326      1b28:       27bd0030        addiu   sp,sp,48
1327      1b2c:       03e00008        jr      ra
1328      1b30:       00000000        nop
1329          ...
1330
1331  00001b40 <create_file>:
1332  #include "file.h"
1333  #include <stdlib.h>
1334  #include <string.h>
1335  #include <errno.h>
1336
1337  void create_file(File *file) {
```

```
1338     1b40:       27bdfff8        addiu   sp,sp,-8
1339     1b44:       afbe0004        sw      s8,4(sp)
1340     1b48:       03a0f025        move    s8,sp
1341     1b4c:       afc40008        sw      a0,8(s8)
1342     file->file = 0;
1343     1b50:       8fc20008        lw      v0,8(s8)
1344     1b54:       ac400000        sw      zero,0(v0)
1345     file->eof = 0;
1346     1b58:       8fc20008        lw      v0,8(s8)
1347     1b5c:       a0400004        sb      zero,4(v0)
1348 }
1349     1b60:       00000000        nop
1350     1b64:       03c0e825        move    sp,s8
1351     1b68:       8fbe0004        lw      s8,4(sp)
1352     1b6c:       27bd0008        addiu   sp,sp,8
1353     1b70:       03e00008        jr      ra
1354     1b74:       00000000        nop
1355
1356 00001b78 <open_file_read>:
1357
1358 char open_file_read(File *file, const char *route) {
1359     1b78:       3c1c0002        lui     gp,0x2
1360     1b7c:       279c8cf8        addiu   gp,gp,-29448
1361     1b80:       0399e021        addu    gp,gp,t9
1362     1b84:       27bdffd0        addiu   sp,sp,-48
1363     1b88:       afbf002c        sw      ra,44(sp)
1364     1b8c:       afbe0028        sw      s8,40(sp)
1365     1b90:       afb00024        sw      s0,36(sp)
1366     1b94:       03a0f025        move    s8,sp
1367     1b98:       afbc0010        sw      gp,16(sp)
1368     1b9c:       afc40030        sw      a0,48(s8)
1369     1ba0:       afc50034        sw      a1,52(s8)
1370     if (route == NULL) {
1371     1ba4:       8fc20034        lw      v0,52(s8)
1372     1ba8:       14400007        bnez    v0,1bc8 <open_file_read+0x50>
1373     1bac:       00000000        nop
1374         file->file = stdin;
1375     1bb0:       8f8280c4        lw      v0,-32572(gp)
1376     1bb4:       8c430000        lw      v1,0(v0)
1377     1bb8:       8fc20030        lw      v0,48(s8)
1378     1bbc:       ac430000        sw      v1,0(v0)
1379     1bc0:       1000002b        b       1c70 <open_file_read+0xf8>
1380     1bc4:       00000000        nop
1381     } else {
1382         file->file = fopen(route, "rb");
1383     1bc8:       8f828030        lw      v0,-32720(gp)
1384     1bcc:       244526c0        addiu   a1,v0,9920
1385     1bd0:       8fc40034        lw      a0,52(s8)
1386     1bd4:       8f8280f4        lw      v0,-32524(gp)
1387     1bd8:       0040c825        move    t9,v0
1388     1bdc:       0320f809        jalr    t9
1389     1be0:       00000000        nop
1390     1be4:       8fdc0010        lw      gp,16(s8)
1391     1be8:       00401825        move    v1,v0
1392     1bec:       8fc20030        lw      v0,48(s8)
1393     1bf0:       ac430000        sw      v1,0(v0)
1394         if (file->file == NULL) {
1395     1bf4:       8fc20030        lw      v0,48(s8)
1396     1bf8:       8c420000        lw      v0,0(v0)
1397     1bfc:       1440001c        bnez    v0,1c70 <open_file_read+0xf8>
1398     1c00:       00000000        nop
1399             int err = errno;
1400     1c04:       8f8280bc        lw      v0,-32580(gp)
1401     1c08:       0040c825        move    t9,v0
1402     1c0c:       0320f809        jalr    t9
1403     1c10:       00000000        nop
1404     1c14:       8fdc0010        lw      gp,16(s8)
```

```
    1c18:       8c420000        lw      v0,0(v0)
    1c1c:       afc20018        sw      v0,24(s8)
            fprintf(stderr, "Error al abrir archivo: %s\n", strerror(err));
    1c20:       8f8280f0        lw      v0,-32528(gp)
    1c24:       8c500000        lw      s0,0(v0)
    1c28:       8fc40018        lw      a0,24(s8)
    1c2c:       8f8280dc        lw      v0,-32548(gp)
    1c30:       0040c825        move    t9,v0
    1c34:       0320f809        jalr    t9
    1c38:       00000000        nop
    1c3c:       8fdc0010        lw      gp,16(s8)
    1c40:       00403025        move    a2,v0
    1c44:       8f828030        lw      v0,-32720(gp)
    1c48:       244526c4        addiu   a1,v0,9924
    1c4c:       02002025        move    a0,s0
    1c50:       8f8280c8        lw      v0,-32568(gp)
    1c54:       0040c825        move    t9,v0
    1c58:       0320f809        jalr    t9
    1c5c:       00000000        nop
    1c60:       8fdc0010        lw      gp,16(s8)
            return ERROR;
    1c64:       24020001        li      v0,1
    1c68:       10000002        b       1c74 <open_file_read+0xfc>
    1c6c:       00000000        nop
        }
    }
    return OK;
    1c70:       00001025        move    v0,zero
}
    1c74:       03c0e825        move    sp,s8
    1c78:       8fbf002c        lw      ra,44(sp)
    1c7c:       8fbe0028        lw      s8,40(sp)
    1c80:       8fb00024        lw      s0,36(sp)
    1c84:       27bd0030        addiu   sp,sp,48
    1c88:       03e00008        jr      ra
    1c8c:       00000000        nop

00001c90 <open_file_write>:


char open_file_write(File *file, const char *route) {
    1c90:       3c1c0002        lui     gp,0x2
    1c94:       279c8be0        addiu   gp,gp,-29728
    1c98:       0399e021        addu    gp,gp,t9
    1c9c:       27bdffd0        addiu   sp,sp,-48
    1ca0:       afbf002c        sw      ra,44(sp)
    1ca4:       afbe0028        sw      s8,40(sp)
    1ca8:       afb00024        sw      s0,36(sp)
    1cac:       03a0f025        move    s8,sp
    1cb0:       afbc0010        sw      gp,16(sp)
    1cb4:       afc40030        sw      a0,48(s8)
    1cb8:       afc50034        sw      a1,52(s8)
    if (route == NULL) {
    1cbc:       8fc20034        lw      v0,52(s8)
    1cc0:       14400007        bnez    v0,1ce0 <open_file_write+0x50>
    1cc4:       00000000        nop
        file->file = stdout;
    1cc8:       8f8280b8        lw      v0,-32584(gp)
    1ccc:       8c430000        lw      v1,0(v0)
    1cd0:       8fc20030        lw      v0,48(s8)
    1cd4:       ac430000        sw      v1,0(v0)
    1cd8:       1000002b        b       1d88 <open_file_write+0xf8>
    1cdc:       00000000        nop
    } else {
        file->file = fopen(route, "wb");
    1ce0:       8f828030        lw      v0,-32720(gp)
    1ce4:       244526e0        addiu   a1,v0,9952
```

```
1472      1ce8:       8fc40034        lw      a0,52(s8)
1473      1cec:       8f8280f4        lw      v0,-32524(gp)
1474      1cf0:       0040c825        move    t9,v0
1475      1cf4:       0320f809        jalr    t9
1476      1cf8:       00000000        nop
1477      1cfc:       8fdc0010        lw      gp,16(s8)
1478      1d00:       00401825        move    v1,v0
1479      1d04:       8fc20030        lw      v0,48(s8)
1480      1d08:       ac430000        sw      v1,0(v0)
1481          if (file->file == NULL) {
1482      1d0c:       8fc20030        lw      v0,48(s8)
1483      1d10:       8c420000        lw      v0,0(v0)
1484      1d14:       1440001c        bnez    v0,1d88 <open_file_write+0xf8>
1485      1d18:       00000000        nop
1486              int err = errno;
1487      1d1c:       8f8280bc        lw      v0,-32580(gp)
1488      1d20:       0040c825        move    t9,v0
1489      1d24:       0320f809        jalr    t9
1490      1d28:       00000000        nop
1491      1d2c:       8fdc0010        lw      gp,16(s8)
1492      1d30:       8c420000        lw      v0,0(v0)
1493      1d34:       afc20018        sw      v0,24(s8)
1494              fprintf(stderr, "Error al abrir archivo: %s\n", strerror(err));
1495      1d38:       8f8280f0        lw      v0,-32528(gp)
1496      1d3c:       8c500000        lw      s0,0(v0)
1497      1d40:       8fc40018        lw      a0,24(s8)
1498      1d44:       8f8280dc        lw      v0,-32548(gp)
1499      1d48:       0040c825        move    t9,v0
1500      1d4c:       0320f809        jalr    t9
1501      1d50:       00000000        nop
1502      1d54:       8fdc0010        lw      gp,16(s8)
1503      1d58:       00403025        move    a2,v0
1504      1d5c:       8f828030        lw      v0,-32720(gp)
1505      1d60:       244526c4        addiu   a1,v0,9924
1506      1d64:       02002025        move    a0,s0
1507      1d68:       8f8280c8        lw      v0,-32568(gp)
1508      1d6c:       0040c825        move    t9,v0
1509      1d70:       0320f809        jalr    t9
1510      1d74:       00000000        nop
1511      1d78:       8fdc0010        lw      gp,16(s8)
1512              return ERROR;
1513      1d7c:       24020001        li      v0,1
1514      1d80:       10000002        b       1d8c <open_file_write+0xfc>
1515      1d84:       00000000        nop
1516          }
1517      }
1518      return OK;
1519      1d88:       00001025        move    v0,zero
1520  }
1521      1d8c:       03c0e825        move    sp,s8
1522      1d90:       8fbf002c        lw      ra,44(sp)
1523      1d94:       8fbe0028        lw      s8,40(sp)
1524      1d98:       8fb00024        lw      s0,36(sp)
1525      1d9c:       27bd0030        addiu   sp,sp,48
1526      1da0:       03e00008        jr      ra
1527      1da4:       00000000        nop
1528
1529  00001da8 <close_file>:
1530
1531
1532  int close_file(File *file) {
1533      1da8:       3c1c0002        lui     gp,0x2
1534      1dac:       279c8ac8        addiu   gp,gp,-30008
1535      1db0:       0399e021        addu    gp,gp,t9
1536      1db4:       27bdffd0        addiu   sp,sp,-48
1537      1db8:       afbf002c        sw      ra,44(sp)
1538      1dbc:       afbe0028        sw      s8,40(sp)
```

```
1539    1dc0:       afb00024        sw      s0,36(sp)
1540    1dc4:       03a0f025        move    s8,sp
1541    1dc8:       afbc0010        sw      gp,16(sp)
1542    1dcc:       afc40030        sw      a0,48(s8)
1543    if (file->file == stdin || file->file == stdout) return OK;
1544    1dd0:       8fc20030        lw      v0,48(s8)
1545    1dd4:       8c430000        lw      v1,0(v0)
1546    1dd8:       8f8280c4        lw      v0,-32572(gp)
1547    1ddc:       8c420000        lw      v0,0(v0)
1548    1de0:       10620007        beq     v1,v0,1e00 <close_file+0x58>
1549    1de4:       00000000        nop
1550    1de8:       8fc20030        lw      v0,48(s8)
1551    1dec:       8c430000        lw      v1,0(v0)
1552    1df0:       8f8280b8        lw      v0,-32584(gp)
1553    1df4:       8c420000        lw      v0,0(v0)
1554    1df8:       14620004        bne     v1,v0,1e0c <close_file+0x64>
1555    1dfc:       00000000        nop
1556    1e00:       00001025        move    v0,zero
1557    1e04:       1000002a        b       1eb0 <close_file+0x108>
1558    1e08:       00000000        nop
1559
1560    int result = fclose(file->file);
1561    1e0c:       8fc20030        lw      v0,48(s8)
1562    1e10:       8c420000        lw      v0,0(v0)
1563    1e14:       00402025        move    a0,v0
1564    1e18:       8f8280ac        lw      v0,-32596(gp)
1565    1e1c:       0040c825        move    t9,v0
1566    1e20:       0320f809        jalr    t9
1567    1e24:       00000000        nop
1568    1e28:       8fdc0010        lw      gp,16(s8)
1569    1e2c:       afc20018        sw      v0,24(s8)
1570    if (result == EOF) {
1571    1e30:       8fc30018        lw      v1,24(s8)
1572    1e34:       2402ffff        li      v0,-1
1573    1e38:       1462001c        bne     v1,v0,1eac <close_file+0x104>
1574    1e3c:       00000000        nop
1575        int err = errno;
1576    1e40:       8f8280bc        lw      v0,-32580(gp)
1577    1e44:       0040c825        move    t9,v0
1578    1e48:       0320f809        jalr    t9
1579    1e4c:       00000000        nop
1580    1e50:       8fdc0010        lw      gp,16(s8)
1581    1e54:       8c420000        lw      v0,0(v0)
1582    1e58:       afc2001c        sw      v0,28(s8)
1583        fprintf(stderr, "Error al cerrar archivo: %s\n", strerror(err));
1584    1e5c:       8f8280f0        lw      v0,-32528(gp)
1585    1e60:       8c500000        lw      s0,0(v0)
1586    1e64:       8fc4001c        lw      a0,28(s8)
1587    1e68:       8f8280dc        lw      v0,-32548(gp)
1588    1e6c:       0040c825        move    t9,v0
1589    1e70:       0320f809        jalr    t9
1590    1e74:       00000000        nop
1591    1e78:       8fdc0010        lw      gp,16(s8)
1592    1e7c:       00403025        move    a2,v0
1593    1e80:       8f828030        lw      v0,-32720(gp)
1594    1e84:       244526e4        addiu   a1,v0,9956
1595    1e88:       02002025        move    a0,s0
1596    1e8c:       8f8280c8        lw      v0,-32568(gp)
1597    1e90:       0040c825        move    t9,v0
1598    1e94:       0320f809        jalr    t9
1599    1e98:       00000000        nop
1600    1e9c:       8fdc0010        lw      gp,16(s8)
1601        return ERROR;
1602    1ea0:       24020001        li      v0,1
1603    1ea4:       10000002        b       1eb0 <close_file+0x108>
1604    1ea8:       00000000        nop
1605    }
```

```
1606        return OK;
1607    1eac:       00001025        move    v0,zero
1608    }
1609    1eb0:       03c0e825        move    sp,s8
1610    1eb4:       8fbf002c        lw      ra,44(sp)
1611    1eb8:       8fbe0028        lw      s8,40(sp)
1612    1ebc:       8fb00024        lw      s0,36(sp)
1613    1ec0:       27bd0030        addiu   sp,sp,48
1614    1ec4:       03e00008        jr      ra
1615    1ec8:       00000000        nop
1616
1617    00001ecc <file_read>:
1618
1619
1620    unsigned int file_read(File *file, unsigned char *buffer, unsigned int length) {
1621    1ecc:       3c1c0002        lui     gp,0x2
1622    1ed0:       279c89a4        addiu   gp,gp,-30300
1623    1ed4:       0399e021        addu    gp,gp,t9
1624    1ed8:       27bdffd8        addiu   sp,sp,-40
1625    1edc:       afbf0024        sw      ra,36(sp)
1626    1ee0:       afbe0020        sw      s8,32(sp)
1627    1ee4:       03a0f025        move    s8,sp
1628    1ee8:       afbc0010        sw      gp,16(sp)
1629    1eec:       afc40028        sw      a0,40(s8)
1630    1ef0:       afc5002c        sw      a1,44(s8)
1631    1ef4:       afc60030        sw      a2,48(s8)
1632        unsigned int result = 0;
1633    1ef8:       afc00018        sw      zero,24(s8)
1634        if (!file_eof(file)) {
1635    1efc:       8fc40028        lw      a0,40(s8)
1636    1f00:       8f828058        lw      v0,-32680(gp)
1637    1f04:       0040c825        move    t9,v0
1638    1f08:       04110041        bal     2010 <file_eof>
1639    1f0c:       00000000        nop
1640    1f10:       8fdc0010        lw      gp,16(s8)
1641    1f14:       1440001a        bnez    v0,1f80 <file_read+0xb4>
1642    1f18:       00000000        nop
1643            result = (unsigned int) fread(buffer, sizeof(char), length, file->file);
1644    1f1c:       8fc20028        lw      v0,40(s8)
1645    1f20:       8c420000        lw      v0,0(v0)
1646    1f24:       00403825        move    a3,v0
1647    1f28:       8fc60030        lw      a2,48(s8)
1648    1f2c:       24050001        li      a1,1
1649    1f30:       8fc4002c        lw      a0,44(s8)
1650    1f34:       8f8280e4        lw      v0,-32540(gp)
1651    1f38:       0040c825        move    t9,v0
1652    1f3c:       0320f809        jalr    t9
1653    1f40:       00000000        nop
1654    1f44:       8fdc0010        lw      gp,16(s8)
1655    1f48:       afc20018        sw      v0,24(s8)
1656            if (feof(file->file)) {
1657    1f4c:       8fc20028        lw      v0,40(s8)
1658    1f50:       8c420000        lw      v0,0(v0)
1659    1f54:       00402025        move    a0,v0
1660    1f58:       8f8280d4        lw      v0,-32556(gp)
1661    1f5c:       0040c825        move    t9,v0
1662    1f60:       0320f809        jalr    t9
1663    1f64:       00000000        nop
1664    1f68:       8fdc0010        lw      gp,16(s8)
1665    1f6c:       10400004        beqz    v0,1f80 <file_read+0xb4>
1666    1f70:       00000000        nop
1667                file->eof = 1;
1668    1f74:       8fc20028        lw      v0,40(s8)
1669    1f78:       24030001        li      v1,1
1670    1f7c:       a0430004        sb      v1,4(v0)
1671            }
1672        }
```

```
1673
1674        return result;
1675        1f80:       8fc20018        lw      v0,24(s8)
1676    }
1677        1f84:       03c0e825        move    sp,s8
1678        1f88:       8fbf0024        lw      ra,36(sp)
1679        1f8c:       8fbe0020        lw      s8,32(sp)
1680        1f90:       27bd0028        addiu   sp,sp,40
1681        1f94:       03e00008        jr      ra
1682        1f98:       00000000        nop
1683
1684    00001f9c <file_write>:
1685
1686
1687    void file_write(File *file, unsigned char *buffer, unsigned int length) {
1688        1f9c:       3c1c0002        lui     gp,0x2
1689        1fa0:       279c88d4        addiu   gp,gp,-30508
1690        1fa4:       0399e021        addu    gp,gp,t9
1691        1fa8:       27bdffe0        addiu   sp,sp,-32
1692        1fac:       afbf001c        sw      ra,28(sp)
1693        1fb0:       afbe0018        sw      s8,24(sp)
1694        1fb4:       03a0f025        move    s8,sp
1695        1fb8:       afbc0010        sw      gp,16(sp)
1696        1fbc:       afc40020        sw      a0,32(s8)
1697        1fc0:       afc50024        sw      a1,36(s8)
1698        1fc4:       afc60028        sw      a2,40(s8)
1699        fwrite(buffer, sizeof(char), length, file->file);
1700        1fc8:       8fc20020        lw      v0,32(s8)
1701        1fcc:       8c420000        lw      v0,0(v0)
1702        1fd0:       00403825        move    a3,v0
1703        1fd4:       8fc60028        lw      a2,40(s8)
1704        1fd8:       24050001        li      a1,1
1705        1fdc:       8fc40024        lw      a0,36(s8)
1706        1fe0:       8f8280e8        lw      v0,-32536(gp)
1707        1fe4:       0040c825        move    t9,v0
1708        1fe8:       0320f809        jalr    t9
1709        1fec:       00000000        nop
1710        1ff0:       8fdc0010        lw      gp,16(s8)
1711    }
1712        1ff4:       00000000        nop
1713        1ff8:       03c0e825        move    sp,s8
1714        1ffc:       8fbf001c        lw      ra,28(sp)
1715        2000:       8fbe0018        lw      s8,24(sp)
1716        2004:       27bd0020        addiu   sp,sp,32
1717        2008:       03e00008        jr      ra
1718        200c:       00000000        nop
1719
1720    00002010 <file_eof>:
1721
1722
1723    int file_eof(File *file) {
1724        2010:       27bdfff8        addiu   sp,sp,-8
1725        2014:       afbe0004        sw      s8,4(sp)
1726        2018:       03a0f025        move    s8,sp
1727        201c:       afc40008        sw      a0,8(s8)
1728        return file->eof;
1729        2020:       8fc20008        lw      v0,8(s8)
1730        2024:       80420004        lb      v0,4(v0)
1731    }
1732        2028:       03c0e825        move    sp,s8
1733        202c:       8fbe0004        lw      s8,4(sp)
1734        2030:       27bd0008        addiu   sp,sp,8
1735        2034:       03e00008        jr      ra
1736        2038:       00000000        nop
1737        203c:       00000000        nop
1738
1739    00002040 <main>:
```

```
#include <stddef.h>
#include <getopt.h>
#include "constants.h"
#include "command.h"

int main(int argc, char **argv) {
    2040:       3c1c0002        lui     gp,0x2
    2044:       279c8830        addiu   gp,gp,-30672
    2048:       0399e021        addu    gp,gp,t9
    204c:       27bdff58        addiu   sp,sp,-168
    2050:       afbf00a4        sw      ra,164(sp)
    2054:       afbe00a0        sw      s8,160(sp)
    2058:       03a0f025        move    s8,sp
    205c:       afbc0018        sw      gp,24(sp)
    2060:       afc400a8        sw      a0,168(s8)
    2064:       afc500ac        sw      a1,172(s8)
    struct option arg_long[] = {
    2068:       8f828028        lw      v0,-32728(gp)
    206c:       27c30028        addiu   v1,s8,40
    2070:       24422820        addiu   v0,v0,10272
    2074:       24040050        li      a0,80
    2078:       00803025        move    a2,a0
    207c:       00402825        move    a1,v0
    2080:       00602025        move    a0,v1
    2084:       8f8280ec        lw      v0,-32532(gp)
    2088:       0040c825        move    t9,v0
    208c:       0320f809        jalr    t9
    2090:       00000000        nop
    2094:       8fdc0018        lw      gp,24(s8)
            {"decode",  no_argument, NULL, 'd'},
            {"help",    no_argument, NULL, 'h'},
            {"version", no_argument, NULL, 'V'},
    };

    char arg_opt_str[] = "i:o:dhV";
    2098:       8f828030        lw      v0,-32720(gp)
    209c:       8c432738        lw      v1,10040(v0)
    20a0:       24422738        addiu   v0,v0,10040
    20a4:       8c420004        lw      v0,4(v0)
    20a8:       afc30078        sw      v1,120(s8)
    20ac:       afc2007c        sw      v0,124(s8)
    int arg_opt;
    int arg_opt_idx = 0;
    20b0:       afc00080        sw      zero,128(s8)

    command_options_st cmd_options;
    // Default Values: encode, stdin as input, stdout as output, stderr as error output
    command_create(&cmd_options);
    20b4:       27c20084        addiu   v0,s8,132
    20b8:       00402025        move    a0,v0
    20bc:       8f828070        lw      v0,-32656(gp)
    20c0:       0040c825        move    t9,v0
    20c4:       0411fb1a        bal     d30 <command_create>
    20c8:       00000000        nop
    20cc:       8fdc0018        lw      gp,24(s8)

    char should_process = TRUE;
    20d0:       24020001        li      v0,1
    20d4:       a3c20020        sb      v0,32(s8)
    while ((arg_opt = getopt_long(argc, argv, arg_opt_str, arg_long, &arg_opt_idx)) != -1
    20d8:       10000048        b       21fc <main+0x1bc>
    20dc:       00000000        nop
           &&
           should_process) {

        switch (arg_opt) {
    20e0:       8fc20024        lw      v0,36(s8)
```

```
1807    20e4:       2442ffaa         addiu   v0,v0,-86
1808    20e8:       2c43001a         sltiu   v1,v0,26
1809    20ec:       1060003a         beqz    v1,21d8 <main+0x198>
1810    20f0:       00000000         nop
1811    20f4:       00021880         sll     v1,v0,0x2
1812    20f8:       8f828030         lw      v0,-32720(gp)
1813    20fc:       24422740         addiu   v0,v0,10048
1814    2100:       00621021         addu    v0,v1,v0
1815    2104:       8c420000         lw      v0,0(v0)
1816    2108:       005c1021         addu    v0,v0,gp
1817    210c:       00400008         jr      v0
1818    2110:       00000000         nop
1819            case 'i': {
1820                // Set Input File
1821                set_input_file(&cmd_options, optarg);
1822    2114:       8f8280a4         lw      v0,-32604(gp)
1823    2118:       8c430000         lw      v1,0(v0)
1824    211c:       27c20084         addiu   v0,s8,132
1825    2120:       00602825         move    a1,v1
1826    2124:       00402025         move    a0,v0
1827    2128:       8f828074         lw      v0,-32652(gp)
1828    212c:       0040c825         move    t9,v0
1829    2130:       0411fb27         bal     dd0 <set_input_file>
1830    2134:       00000000         nop
1831    2138:       8fdc0018         lw      gp,24(s8)
1832            }
1833                break;
1834    213c:       1000002f         b       21fc <main+0x1bc>
1835    2140:       00000000         nop
1836            case 'o': {
1837                // Set Output File
1838                set_output_file(&cmd_options, optarg);
1839    2144:       8f8280a4         lw      v0,-32604(gp)
1840    2148:       8c430000         lw      v1,0(v0)
1841    214c:       27c20084         addiu   v0,s8,132
1842    2150:       00602825         move    a1,v1
1843    2154:       00402025         move    a0,v0
1844    2158:       8f828078         lw      v0,-32648(gp)
1845    215c:       0040c825         move    t9,v0
1846    2160:       0411fb29         bal     e08 <set_output_file>
1847    2164:       00000000         nop
1848    2168:       8fdc0018         lw      gp,24(s8)
1849            }
1850                break;
1851    216c:       10000023         b       21fc <main+0x1bc>
1852    2170:       00000000         nop
1853            case 'h': {
1854                // Show Options
1855                show_help();
1856    2174:       8f828038         lw      v0,-32712(gp)
1857    2178:       0040c825         move    t9,v0
1858    217c:       0411fbb6         bal     1058 <show_help>
1859    2180:       00000000         nop
1860    2184:       8fdc0018         lw      gp,24(s8)
1861                should_process = FALSE;
1862    2188:       a3c00020         sb      zero,32(s8)
1863            }
1864                break;
1865    218c:       1000001b         b       21fc <main+0x1bc>
1866    2190:       00000000         nop
1867            case 'V': {
1868                // Show Version
1869                show_version();
1870    2194:       8f82807c         lw      v0,-32644(gp)
1871    2198:       0040c825         move    t9,v0
1872    219c:       0411fbfc         bal     1190 <show_version>
1873    21a0:       00000000         nop
```

```
1874    21a4:       8fdc0018        lw      gp,24(s8)
1875                should_process = FALSE;
1876    21a8:       a3c00020        sb      zero,32(s8)
1877            }
1878                break;
1879    21ac:       10000013        b       21fc <main+0x1bc>
1880    21b0:       00000000        nop
1881            case 'd': {
1882                // Set Decode option
1883                set_decode(&cmd_options);
1884    21b4:       27c20084        addiu   v0,s8,132
1885    21b8:       00402025        move    a0,v0
1886    21bc:       8f828080        lw      v0,-32640(gp)
1887    21c0:       0040c825        move    t9,v0
1888    21c4:       0411fb2b        bal     e74 <set_decode>
1889    21c8:       00000000        nop
1890    21cc:       8fdc0018        lw      gp,24(s8)
1891            }
1892                break;
1893    21d0:       1000000a        b       21fc <main+0x1bc>
1894    21d4:       00000000        nop
1895            default: {
1896                // Set Error Condition = INVALID_ARGUMENT
1897                // [Gonzalo: We analize only valid arguments values here: i, o, h, V, d]
1898                set_error(&cmd_options, INVALID_ARGUMENT);
1899    21d8:       27c20084        addiu   v0,s8,132
1900    21dc:       2405ffff        li      a1,-1
1901    21e0:       00402025        move    a0,v0
1902    21e4:       8f82805c        lw      v0,-32676(gp)
1903    21e8:       0040c825        move    t9,v0
1904    21ec:       0411fb2d        bal     ea4 <set_error>
1905    21f0:       00000000        nop
1906    21f4:       8fdc0018        lw      gp,24(s8)
1907            }
1908                break;
1909    21f8:       00000000        nop
1910    while ((arg_opt = getopt_long(argc, argv, arg_opt_str, arg_long, &arg_opt_idx)) != -1
1911    21fc:       27c40028        addiu   a0,s8,40
1912    2200:       27c30078        addiu   v1,s8,120
1913    2204:       27c20080        addiu   v0,s8,128
1914    2208:       afa20010        sw      v0,16(sp)
1915    220c:       00803825        move    a3,a0
1916    2210:       00603025        move    a2,v1
1917    2214:       8fc500ac        lw      a1,172(s8)
1918    2218:       8fc400a8        lw      a0,168(s8)
1919    221c:       8f8280d0        lw      v0,-32560(gp)
1920    2220:       0040c825        move    t9,v0
1921    2224:       0320f809        jalr    t9
1922    2228:       00000000        nop
1923    222c:       8fdc0018        lw      gp,24(s8)
1924    2230:       afc20024        sw      v0,36(s8)
1925    2234:       8fc30024        lw      v1,36(s8)
1926    2238:       2402ffff        li      v0,-1
1927    223c:       10620004        beq     v1,v0,2250 <main+0x210>
1928    2240:       00000000        nop
1929           &&
1930    2244:       83c20020        lb      v0,32(s8)
1931    2248:       1440ffa5        bnez    v0,20e0 <main+0xa0>
1932    224c:       00000000        nop
1933        }
1934    }
1935
1936    // Help or Version arguments, no processing
1937    if (!should_process) {
1938    2250:       83c20020        lb      v0,32(s8)
1939    2254:       14400004        bnez    v0,2268 <main+0x228>
1940    2258:       00000000        nop
```

```
1941            return 0;
1942        225c:       00001025        move    v0,zero
1943        2260:       1000001b        b       22d0 <main+0x290>
1944        2264:       00000000        nop
1945        }
1946
1947        // Processs Encode/Decode if no errors found, otherwise show error message
1948        if (has_errors(&cmd_options)) {
1949        2268:       27c20084        addiu   v0,s8,132
1950        226c:       00402025        move    a0,v0
1951        2270:       8f828068        lw      v0,-32664(gp)
1952        2274:       0040c825        move    t9,v0
1953        2278:       0411fb19        bal     ee0 <has_errors>
1954        227c:       00000000        nop
1955        2280:       8fdc0018        lw      gp,24(s8)
1956        2284:       1040000b        beqz    v0,22b4 <main+0x274>
1957        2288:       00000000        nop
1958            show_error(&cmd_options);
1959        228c:       27c20084        addiu   v0,s8,132
1960        2290:       00402025        move    a0,v0
1961        2294:       8f828060        lw      v0,-32672(gp)
1962        2298:       0040c825        move    t9,v0
1963        229c:       0411fb21        bal     f24 <show_error>
1964        22a0:       00000000        nop
1965        22a4:       8fdc0018        lw      gp,24(s8)
1966            return 1;
1967        22a8:       24020001        li      v0,1
1968        22ac:       10000008        b       22d0 <main+0x290>
1969        22b0:       00000000        nop
1970        }
1971
1972        return process(&cmd_options);
1973        22b4:       27c20084        addiu   v0,s8,132
1974        22b8:       00402025        move    a0,v0
1975        22bc:       8f828084        lw      v0,-32636(gp)
1976        22c0:       0040c825        move    t9,v0
1977        22c4:       0411fbc8        bal     11e8 <process>
1978        22c8:       00000000        nop
1979        22cc:       8fdc0018        lw      gp,24(s8)
1980    }
1981        22d0:       03c0e825        move    sp,s8
1982        22d4:       8fbf00a4        lw      ra,164(sp)
1983        22d8:       8fbe00a0        lw      s8,160(sp)
1984        22dc:       27bd00a8        addiu   sp,sp,168
1985        22e0:       03e00008        jr      ra
1986        22e4:       00000000        nop
1987            ...
1988
1989    000022f0 <__libc_csu_init>:
1990        22f0:       3c1c0002        lui     gp,0x2
1991        22f4:       279c8580        addiu   gp,gp,-31360
1992        22f8:       0399e021        addu    gp,gp,t9
1993        22fc:       27bdffc8        addiu   sp,sp,-56
1994        2300:       8f998088        lw      t9,-32632(gp)
1995        2304:       afbc0010        sw      gp,16(sp)
1996        2308:       afb50030        sw      s5,48(sp)
1997        230c:       00c0a825        move    s5,a2
1998        2310:       afb4002c        sw      s4,44(sp)
1999        2314:       00a0a025        move    s4,a1
2000        2318:       afb30028        sw      s3,40(sp)
2001        231c:       00809825        move    s3,a0
2002        2320:       afb20024        sw      s2,36(sp)
2003        2324:       afb0001c        sw      s0,28(sp)
2004        2328:       afbf0034        sw      ra,52(sp)
2005        232c:       0411f9fe        bal     b28 <_init>
2006        2330:       afb10020        sw      s1,32(sp)
2007        2334:       8fbc0010        lw      gp,16(sp)
```

```
    2338:       8f90808c        lw      s0,-32628(gp)
    233c:       8f928090        lw      s2,-32624(gp)
    2340:       02509023        subu    s2,s2,s0
    2344:       00129083        sra     s2,s2,0x2
    2348:       12400009        beqz    s2,2370 <__libc_csu_init+0x80>
    234c:       00008825        move    s1,zero
    2350:       8e190000        lw      t9,0(s0)
    2354:       26310001        addiu   s1,s1,1
    2358:       02a03025        move    a2,s5
    235c:       02802825        move    a1,s4
    2360:       0320f809        jalr    t9
    2364:       02602025        move    a0,s3
    2368:       1651fff9        bne     s2,s1,2350 <__libc_csu_init+0x60>
    236c:       26100004        addiu   s0,s0,4
    2370:       8fbf0034        lw      ra,52(sp)
    2374:       8fb50030        lw      s5,48(sp)
    2378:       8fb4002c        lw      s4,44(sp)
    237c:       8fb30028        lw      s3,40(sp)
    2380:       8fb20024        lw      s2,36(sp)
    2384:       8fb10020        lw      s1,32(sp)
    2388:       8fb0001c        lw      s0,28(sp)
    238c:       03e00008        jr      ra
    2390:       27bd0038        addiu   sp,sp,56

00002394 <__libc_csu_fini>:
    2394:       03e00008        jr      ra
    2398:       00000000        nop
    239c:       00000000        nop

Disassembly of section .MIPS.stubs:

000023a0 <_MIPS_STUBS_>:
    23a0:       8f998010        lw      t9,-32752(gp)
    23a4:       03e07825        move    t7,ra
    23a8:       0320f809        jalr    t9
    23ac:       24180033        li      t8,51
    23b0:       8f998010        lw      t9,-32752(gp)
    23b4:       03e07825        move    t7,ra
    23b8:       0320f809        jalr    t9
    23bc:       24180031        li      t8,49
    23c0:       8f998010        lw      t9,-32752(gp)
    23c4:       03e07825        move    t7,ra
    23c8:       0320f809        jalr    t9
    23cc:       24180030        li      t8,48
    23d0:       8f998010        lw      t9,-32752(gp)
    23d4:       03e07825        move    t7,ra
    23d8:       0320f809        jalr    t9
    23dc:       2418002f        li      t8,47
    23e0:       8f998010        lw      t9,-32752(gp)
    23e4:       03e07825        move    t7,ra
    23e8:       0320f809        jalr    t9
    23ec:       2418002e        li      t8,46
    23f0:       8f998010        lw      t9,-32752(gp)
    23f4:       03e07825        move    t7,ra
    23f8:       0320f809        jalr    t9
    23fc:       2418002d        li      t8,45
    2400:       8f998010        lw      t9,-32752(gp)
    2404:       03e07825        move    t7,ra
    2408:       0320f809        jalr    t9
    240c:       2418002b        li      t8,43
    2410:       8f998010        lw      t9,-32752(gp)
    2414:       03e07825        move    t7,ra
    2418:       0320f809        jalr    t9
    241c:       2418002a        li      t8,42
    2420:       8f998010        lw      t9,-32752(gp)
    2424:       03e07825        move    t7,ra
    2428:       0320f809        jalr    t9
```

```
    242c:       24180029        li      t8,41
    2430:       8f998010        lw      t9,-32752(gp)
    2434:       03e07825        move    t7,ra
    2438:       0320f809        jalr    t9
    243c:       24180028        li      t8,40
    2440:       8f998010        lw      t9,-32752(gp)
    2444:       03e07825        move    t7,ra
    2448:       0320f809        jalr    t9
    244c:       24180026        li      t8,38
    2450:       8f998010        lw      t9,-32752(gp)
    2454:       03e07825        move    t7,ra
    2458:       0320f809        jalr    t9
    245c:       24180025        li      t8,37
    2460:       8f998010        lw      t9,-32752(gp)
    2464:       03e07825        move    t7,ra
    2468:       0320f809        jalr    t9
    246c:       24180023        li      t8,35
    2470:       8f998010        lw      t9,-32752(gp)
    2474:       03e07825        move    t7,ra
    2478:       0320f809        jalr    t9
    247c:       24180021        li      t8,33
    2480:       8f998010        lw      t9,-32752(gp)
    2484:       03e07825        move    t7,ra
    2488:       0320f809        jalr    t9
    248c:       2418001e        li      t8,30
        ...

Disassembly of section .fini:

000024a0 <_fini>:
    24a0:       3c1c0002        lui     gp,0x2
    24a4:       279c83d0        addiu   gp,gp,-31792
    24a8:       0399e021        addu    gp,gp,t9
    24ac:       27bdffe0        addiu   sp,sp,-32
    24b0:       afbc0010        sw      gp,16(sp)
    24b4:       afbf001c        sw      ra,28(sp)
    24b8:       8fbf001c        lw      ra,28(sp)
    24bc:       03e00008        jr      ra
    24c0:       27bd0020        addiu   sp,sp,32
```