FACULTAD DE INGENIERIA
Universidad de Buenos Aires

TP0 - Infraestructura básica

[66.20] Organización del Computador
Segundo cuatrimestre de 2020

Mariotti, Franco	102223	fmariotti@fi.uba.ar	Franco Mariotti
Fabbiano, Fernando	102464	ffabbiano@fi.uba.ar	Fernado Fabbiano
Alasino, Franco Federico	102165	falasino@fi.uba.ar	Franco Alasino

Índice

1. Introducción

En el presente trabajo, además de familiarizarnos con el entorno de desarrollo a utilizar durante la cursada, se desarrollo un programa que permite convertir archivos de texto a archivos en base64. A su vez, este mismo programa nos permite hacer el camino inverso, realizando una decodificación para pasar de base64 a texto.

Además de lo anteriormente mencionado, se utilizan los descriptores de archivo: `stdin, stdout y stderr`. El primero correspondiente a la entrada estándar, el segundo a la salida estándar, y el último correspondiente al error estándar.

Por último, se llevo a cabo una investigación y posterior utilización de la función `'getopt long'` para el manejo de las distintas opciones que se le brindan al usuario para interactuar con el programa en cuestión.

2. Diseño e Implementación

El programa consiste en un módulo principal `-main-` que se encarga de handlear las distintas opciones que se le brindan al usuario, sabiendo como responder a los comandos requeridos en el enunciado, pero también a casos en el que el usuario quiere vulnerar el sistema ingresando, por ejemplo, una opción inválida. Esta función main la podemos encontrar en el archivo `'tp0.c'`, junto con una función para abrir, leer, escribir y cerrar los archivos correspondientes.

Luego tenemos el archivo `'utils.c'` el cual contiene las funciones para realizar la codificación o decodificación de los textos adjuntados.

Por último, el archivo `'utils.h'` contiene la declaración de los métodos implementados en el archivo mencionado mencionado inmediatamente antes.

3. Proceso de Compilación

Para la compilación del programa, basta con ingresar en una consola, en el directorio correspondiente al proyecto, el comando `'make'`. Esto generará automáticamente los archivos binarios correspondientes, y los ejecutables (tanto el principal, como el de pruebas). Este último se guarda con el nombre de `'tp0'`.

Para la ejecución y posterior utilización del programa, se debe ingresar `'./tp0 -comando'` donde `'comando'` puede ser una de la siguientes opciones:

- -V, - -version –>Print version and quit
- -h, - -help –>Print información de ayuda acerca de los comandos
- -o, - -output –>Path para el output file donde se escribirá la salida del programa
- -i, - -input –>Path para el input file que se desea codificar a base 64
- -d, - -decode –>Decodificar un archivo encodeado en base64

4. Casos de Prueba

A continuación se muestra la salida por consola que se genera luego de ejecutar el paquete de tests pensado para este trabajo. Para generar dicha salida, será necesario correr el comando `'./test_utils'`. El código de las mismas se encuentra en la sección Código Fuente, en lenguaje C

Figura 1: Salida por consola luego de ejecutar las pruebas correspondientes.

5. Conclusiones

Con este trabajo pudimos entender como funciona y se configura un emulador dentro de nuestro mismo computador. A través de las problemáticas que surgieron a partir de estas instalaciones, fuimos capaces de indagar más y entender, por ejemplo, cómo se hace un túnel entre una máquina host y una guest.

Por otro lado, yendo al núcleo del trabajo, fuimos capaces de entender como funciona el sistema de base 64, y lograr llevarlo a código C. Para esto manipulamos arrays de caracteres a bajo nivel, y usamos, por ejemplo, funciones de desplazamiento de bits (o shifteos) para conseguir lo necesario para codificar y decodificar. Esto último nos parecio muy interesante, dado que hasta este momento de la carrera no habíamos utilizado en detalle dichos operandos en C, y por tanto no estabamos tan familiarizados con los mismos. De alguna forma lo pudimos relacionar con la aplicación de conocimientos mas bien teóricos vistos en la materia Estructura del Computador.

6. Referencias

- base64decode.org –>para codificar y decodificar, usado para comprobar los valores de referencia utilizados en las pruebas
- Wikipedia –>información general sobre el sistema de base64
- mycplusplus –>información acerca del algoritmo de decode y encode del sistema base64.
- Repositrio tp0 –>repositorio de github donde se puede encontrar el código fuente, el presente informe, imágenes y demás recursos utilizados para desarrollar el trabajo. Dicho repositorio es privado, por lo que si se desea consultar algo, pedir acceso.

7. Apéndices

7.1. Código Fuente, en lenguaje C

tp0.c

```c
#define  _POSIX_C_SOURCE 200809L

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

```c
#include <string.h>
#include <getopt.h>
#include <stdbool.h>
#include "utils.h"

#define V_OPTION 'V'
#define H_OPTION 'h'
#define O_OPTION 'o'
#define I_OPTION 'i'
#define D_OPTION 'd'

#define MAX_LONGITUD 256
#define HELP_MESSAGE "Options:\n-V, --version\tPrint version and quit.\n-h, --help\t\
Print this information.\n-o, --output\tPath to output file.\n-i, \
--input\tPath to input file.\n-d, --decode\tDecode a base64-encoded\
file.\n"

#define INVALID_MESSAGE "Invalid option , use -h or --help to list valid commands\n"

enum operacion {DECODE,ENCODE,HELP,VERSION};

void imprimir_salida(const char* mensaje) {
  fprintf(stdout,"%s\n",mensaje);
}

void imprimir_error(const char* error) {
  fprintf(stderr,"%s\n",error);
}

int main(int argc, char **argv){
  int c;
  int operacion = ENCODE;
  const char* inputFileptr = NULL;
  const char* outputFileptr = NULL;
  char inputFileName[MAX_LONGITUD];
  char outputFileName[MAX_LONGITUD];
  memset(inputFileName,0,MAX_LONGITUD);
  memset(outputFileName,0,MAX_LONGITUD);

  while (true) {
    int option_index = 0;

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

    c = getopt_long(argc, argv, "Vhdi:o:", long_options, &option_index);
    if (c == -1)
      break;

```

```
60      switch (c) {
61        case V_OPTION:
62          operacion = VERSION;
63          break;
64        case H_OPTION:
65          operacion = HELP;
66          break;
67        case I_OPTION:
68          memcpy(inputFileName,optarg,strlen(optarg));
69          inputFileptr = inputFileName;
70          break;
71        case O_OPTION:
72          memcpy(outputFileName,optarg,strlen(optarg));
73          outputFileptr = outputFileName;
74          break;
75        case D_OPTION:
76          operacion = DECODE;
77          break;
78
79        default:
80          imprimir_salida(INVALID_MESSAGE);
81      }
82    }
83
84    if(operacion == ENCODE || operacion == DECODE) {
85      FILE* outfd = stdout;
86      FILE* infd = stdin;
87
88      if (inputFileptr) {
89        infd = fopen(inputFileName,"r");
90        if (!infd) {
91          imprimir_error("El archivo de input no existe\n");
92          return EXIT_FAILURE;
93        }
94      }
95
96      if (outputFileptr) {
97        outfd = fopen(outputFileName,"w");
98      }
99
100     if(operacion) {
101       encodeFileToBase64(infd,outfd);
102     } else {
103       decodeFileFromBase64(infd,outfd);
104     }
105
106     fclose(infd);
107     fclose(outfd);
108   } else if (operacion == HELP) {
109     imprimir_salida(HELP_MESSAGE);
110   } else if(operacion == VERSION) {
111     imprimir_salida("Version: 1.0.0\n");
112   }
113
```

```
114   return EXIT_SUCCESS;
115 }
```

utils.h

```
1  #ifndef UTILS_H
2  #define UTILS_H
3  #include <stdio.h>
4  #include <stdlib.h>
5  #define ENCODE_SIZE 3 // o un multiplo de 3
6  #define DECODE_SIZE 4 // o un multiplo de 4
7  #define EXIT_FAILURE 1
8  #define SUCCESS 0
9
10 int encodeFileToBase64(FILE* fdin,FILE* fdout);
11 int decodeFileFromBase64(FILE* fdin,FILE* fdout);
12 char* encodeBase64(const char *data,size_t lenInput,size_t* lenOutput);
13 char* decodeBase64(const char *data,size_t lenInput,size_t* lenOutput);
14 #endif
```

utils.c

```
1  #include <stdint.h>
2  #include <string.h>
3  #include <stdbool.h>
4  #include "utils.h"
5
6  #define MASK_SEXTET 0x3F
7  #define MASK_OCTET 0xFF
8  #define SHIFT_SEXTET 6
9  #define BYTE_NULO 0
10 #define CARACTER_IGUAL '='
11
12 static int modTable[] = {0,2,1}; // cant de veces que tiene que iterar
13 static const unsigned char base64_table[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
14 //INICIO FUNCIONES PRIVADAS
15 static bool __b64_isvalidchar(char c) {
16   bool valid = false;
17   if (c >= '0' && c <= '9')
18     valid = true;
19   if (c >= 'A' && c <= 'Z')
20     valid = true;
21   if (c >= 'a' && c <= 'z')
22     valid = true;
23   if (c == '+' || c == '/' || c == '=')
24     valid = true;
25   return valid;
26 }
27
28 static size_t __len_base64_decode_output(const char *input,size_t lenInput) {
29   if (!input) return 0;
30
31   size_t len = 0;
32   len = lenInput / 4 * 3;
33
```

```
34   for (size_t i = 0; i < lenInput; i++) {
35     if (input[i] == CARACTER_IGUAL) {
36       len--;
37     }
38   }
39
40   return len;
41 }
42
43 static size_t __len_base64_encode_output(size_t inputLen) {
44   size_t ouputLen;
45   ouputLen =  inputLen;
46   if (inputLen % 3 != 0)
47     ouputLen += 3 - (inputLen % 3);
48   ouputLen /= 3;
49   ouputLen *= 4;
50
51   return ouputLen;
52 }
53
54 static int __write(FILE* file, char* data, size_t lenData) {
55   if(!file || !data) return EXIT_FAILURE;
56   fwrite(data,sizeof(char),lenData,file);
57   return SUCCESS;
58 }
59
60 static int __processFile(FILE* fdin,FILE* fdout,char* buffer,size_t lenBuffer,bool encode) {
61   char* output = NULL;
62   size_t len = 0;
63   size_t nread = 0;
64   while ((nread = fread(buffer,sizeof(char),lenBuffer,fdin)) > 0) {
65     if(encode) {
66       output = encodeBase64(buffer, nread,&len);
67     } else {
68       output = decodeBase64(buffer, nread,&len);
69     }
70     if(output){
71       __write(fdout,output,len);
72       free(output);
73     }
74     memset(buffer,0,lenBuffer);
75   }
76
77   if(nread == -1) return EXIT_FAILURE;
78
79   return SUCCESS;
80 }
81
82 //FIN FUNCIONES PRIVADAS
83
84 //INICIO FUNCIONES PUBLICAS
85 char* encodeBase64(const char *data,
86                    size_t lenInput,
87                    size_t *lenOutput) {
```

```
88    if(data == NULL || lenInput < 0) return NULL;
89
90    size_t maxlenOutput = __len_base64_encode_output(lenInput);
91    char* output = (char*)calloc(sizeof(char), maxlenOutput + 1);
92    if(!output) return NULL;
93
94    size_t i,j;
95
96    for (i = 0,j=0; i < lenInput; j+=4) {
97      uint32_t octet0 = (unsigned char) data[i++];
98      uint32_t octet1 = i < lenInput ?  data[i++] : BYTE_NULO;
99      uint32_t octet2 = i < lenInput ?  data[i++] : BYTE_NULO;
100
101     uint32_t bits = (octet0 << 16) + (octet1 << 8) + octet2;
102
103     uint32_t sextetA = (bits >> 18) & MASK_SEXTET;
104     uint32_t sextetB = (bits >> 12) & MASK_SEXTET;
105     uint32_t sextetC = (bits >> 6) & MASK_SEXTET;
106     uint32_t sextetD = (bits) & MASK_SEXTET;
107
108     output[j]   = base64_table[sextetA];
109     output[j+1] = base64_table[sextetB];
110     output[j+2] = base64_table[sextetC];
111     output[j+3] = base64_table[sextetD];
112
113   }
114
115   for(int i=0; i < modTable[lenInput%3]; i++)
116     output[maxlenOutput - 1 -i] = CARACTER_IGUAL;
117
118   *lenOutput = strlen(output);
119   return output;
120 }
121
122 char* decodeBase64(const char *data,
123                              size_t lenInput,
124                              size_t *lenOutput) {
125   if (!data) return NULL;
126   unsigned char* dataPtr = (unsigned char*)data;
127
128   unsigned char dTable[256];
129   memset(dTable, -1, 256);
130   for (size_t i = 0; i < sizeof(base64_table) - 1; i++)
131     dTable[base64_table[i]] = (unsigned char) i;
132
133   size_t lenOutputAux = __len_base64_decode_output(data,lenInput);
134
135   char *output = (char*)calloc(sizeof(char), lenOutputAux + 1);
136   if(!output) return NULL;
137
138   uint32_t decode;
139
140   size_t i;
141   size_t j;
```

```
142   for (i=0, j=0; i<lenInput; i+=4, j+=3) {
143     if (!__b64_isvalidchar(data[i])) continue;
144     decode = dTable[dataPtr[i]];
145     decode = (decode << 6) | dTable[dataPtr[i+1]];
146     decode = data[i+2] == CARACTER_IGUAL ? decode << 6 : (decode << 6) | dTable[dataPtr[i+2]];
147     decode = data[i+3] == CARACTER_IGUAL ? decode << 6 : (decode << 6) | dTable[dataPtr[i+3]];
148 
149     output[j] = (decode >> 16) & MASK_OCTET;
150     if (data[i+2] != '=')
151       output[j+1] = (decode >> 8) & MASK_OCTET;
152     if (data[i+3] != '=')
153       output[j+2] = decode & MASK_OCTET;
154   }
155   *lenOutput = strlen(output);
156 
157   return output;
158 }
159 
160 int encodeFileToBase64(FILE* fdin,FILE* fdout) {
161   if (!fdin || !fdout) return EXIT_FAILURE;
162   char buffer[ENCODE_SIZE];
163   memset(buffer,0,ENCODE_SIZE);
164   return __processFile(fdin,fdout,buffer,ENCODE_SIZE,true);
165 }
166 
167 int decodeFileFromBase64(FILE* fdin,FILE* fdout) {
168   if (!fdin || !fdout) return EXIT_FAILURE;
169   char buffer[DECODE_SIZE];
170   memset(buffer,0,DECODE_SIZE);
171   return __processFile(fdin,fdout,buffer,DECODE_SIZE,false);
172 }
173 
174 //FIN FUNCIONES PUBLICAS
```

tests

```
1  #include "utils.h"
2  #include "string.h"
3  
4  #define MAX_LONGITUD 256
5  
6  //Pruebas de la funcion encodeBase64
7  
8  int test00PasoABase64UnaFrase() {
9    size_t outputLen = 0;
10   char* frase = "En un lugar de La Mancha de cuyo nombre no quiero acordarme";
11   char* fraseEnBase64 = "RW4gdW4gbHVnYXIgZGUgTGEgTWFuY2hhIGRlIGN1eW8gbm9tYnJlIG5vIHF1aWVyb byBhY29y
12   char* resultado = encodeBase64(frase,strlen(frase),&outputLen);
13 
14   if (strcmp(fraseEnBase64,resultado) == 0) {
15     printf("Paso una frase a base64 : OK\n");
16     return SUCCESS;
17   } else {
18     printf("Paso una frase a base64 : ERROR\n");
```

```
19      return EXIT_FAILURE;
20    }
21  }
22
23  int test01PasoABase64UnaFraseVacia() {
24    size_t outputLen = 0;
25    char* frase = "";
26    char* resultado = encodeBase64(frase,strlen(frase),&outputLen);
27
28    if (strcmp(frase,resultado) == 0) {
29      printf("Paso una frase vacia a base64 : OK\n");
30      return SUCCESS;
31    } else {
32      printf("Paso una frase vacia a base64 : ERROR\n");
33      return EXIT_FAILURE;
34    }
35  }
36
37  int test02PasoABase64UnaFraseInvalida() {
38    size_t outputLen = 0;
39    char* frase = NULL;
40    char* resultado = encodeBase64(frase,-1,&outputLen);
41
42    if (!resultado) {
43      printf("Paso una frase invalida a base64 : OK\n");
44      return SUCCESS;
45    } else {
46      printf("Paso una frase invalida a base64 : ERROR\n");
47      return EXIT_FAILURE;
48    }
49  }
50
51  //Pruebas de la funcion decodeBase64
52
53  int test00DecodificoUnaFraseEnBase64() {
54    size_t outputLen = 0;
55    char* frase = "En un lugar de La Mancha de cuyo nombre no quiero acordarme";
56    char* fraseEnBase64 = "RW4gdW4gbHVnYXIgZGUgTGEgTWFuY2hhIGRlIGN1eW8gbm9tYnJlIG5vIHF1aWVyby BhY29y
57    char* resultado = decodeBase64(fraseEnBase64,strlen(fraseEnBase64),&outputLen);
58
59
60    if (strcmp(frase,resultado) == 0) {
61      printf("Decodifico frase en base64 : OK\n");
62      return SUCCESS;
63    } else {
64      printf("Decodifico frase en base64 : ERROR\n");
65      return EXIT_FAILURE;
66    }
67  }
68
69  int test01DecodificoUnaFraseVacia() {
70    size_t outputLen = 0;
71    char* frase = "";
72    char* resultado = decodeBase64(frase,strlen(frase),&outputLen);
```

```
73
74    if (strcmp(frase,resultado) == 0) {
75      printf("Decodifico una frase vacia : OK\n");
76      return SUCCESS;
77    } else {
78      printf("Decodifico una frase vacia : ERROR\n");
79      return EXIT_FAILURE;
80    }
81  }
82
83  int test02DecodificoUnaFraseInvalida() {
84    size_t outputLen = 0;
85    char* frase = NULL;
86    char* resultado = decodeBase64(frase,-1,&outputLen);
87
88    if (!resultado) {
89      printf("Decodifico una frase invalida : OK\n");
90      return SUCCESS;
91    } else {
92      printf("Decodifico una frase invalida : ERROR\n");
93      return EXIT_FAILURE;
94    }
95  }
96
97  //Pruebas de la funcion encodeFileToBase64
98
99  int test00CodificoUsandoDosArchivosValidos() {
100   FILE* inputFile = fopen("entrada.txt", "r+");
101   FILE* outputFile = fopen("textoCodificado.txt","r+");
102
103   int resultado = encodeFileToBase64(inputFile,outputFile);
104   char* inputFileEnconding = "RW4gdW4gbHVnYXIgZGUgTGEgTWFuY2hhIGRlIGN1eW8gbm9tYnJlIG5vIHF1aWVyb
105
106   if (resultado == SUCCESS) {
107     char outputFileContent [MAX_LONGITUD];
108     memset(outputFileContent,0,MAX_LONGITUD);
109     fseek(outputFile, 0, SEEK_SET);
110     fread(outputFileContent,sizeof(char),MAX_LONGITUD,outputFile);
111
112     if (strcmp(inputFileEnconding,outputFileContent) == 0) {
113       printf("Codifico usando dos archivos validos : OK\n");
114       fclose(inputFile);
115       fclose(outputFile);
116       return SUCCESS;
117     } else {
118       printf("Codifico usando dos archivos validos : ERROR\n");
119       fclose(inputFile);
120       fclose(outputFile);
121       return EXIT_FAILURE;
122     }
123
124
125   } else {
126     printf("Codifico usando dos archivos validos : ERROR\n");
```

```
127     fclose(inputFile);
128     fclose(outputFile);
129     return EXIT_FAILURE;
130   }
131 }
132
133 int test01CodificoUsandoArchivoDeInputInexistente() {
134   FILE* inputFile = fopen("noExiste.txt", "r+");
135   FILE* outputFile = fopen("textoCodificado.txt","r+");
136
137   int resultado = encodeFileToBase64(inputFile,outputFile);
138
139   if (resultado == EXIT_FAILURE) {
140     printf("Codifico usando archivo de input inexistente : OK\n");
141     fclose(outputFile);
142     return SUCCESS;
143   } else {
144     printf("Codifico usando archivo de input inexistente : ERROR\n");
145     fclose(inputFile);
146     fclose(outputFile);
147     return EXIT_FAILURE;
148   }
149 }
150
151 int test02CodificoUsandoArchivoDeOutputInexistente() {
152   FILE* inputFile = fopen("entrada.txt", "r+");
153   FILE* outputFile = fopen("noExiste.txt","r+");
154
155   int resultado = encodeFileToBase64(inputFile,outputFile);
156
157   if (resultado == EXIT_FAILURE) {
158     printf("Codifico usando archivo de output inexistente : OK\n");
159     fclose(inputFile);
160     return SUCCESS;
161   } else {
162     printf("Codifico usando archivo de output inexistente : ERROR\n");
163     fclose(inputFile);
164     fclose(outputFile);
165     return EXIT_FAILURE;
166   }
167 }
168
169 int test03CodificoUnArchivoBinario() {
170   FILE* inputFile = fopen("pruebaBinario.bin", "r+");
171   FILE* outputFile = fopen("textoBinarioCodificado.bin","r+");
172
173   int resultado = encodeFileToBase64(inputFile,outputFile);
174   char* inputFileEnconding = "YmluYXJpbwo=";
175
176   if (resultado == SUCCESS) {
177     char outputFileContent [MAX_LONGITUD];
178     memset(outputFileContent,0,MAX_LONGITUD);
179     fseek(outputFile, 0, SEEK_SET);
180     fread(outputFileContent,sizeof(char),MAX_LONGITUD,outputFile);
```

```
181
182
183     if (strcmp(inputFileEnconding,outputFileContent) == 0) {
184       printf("Codifico un archivo binario: OK\n");
185       fclose(inputFile);
186       fclose(outputFile);
187       return SUCCESS;
188     } else {
189       printf("Codifico un archivo binario : ERROR\n");
190       fclose(inputFile);
191       fclose(outputFile);
192       return EXIT_FAILURE;
193     }
194
195
196   } else {
197     printf("Codifico un archivo binario : ERROR\n");
198     fclose(inputFile);
199     fclose(outputFile);
200     return EXIT_FAILURE;
201   }
202 }
203
204
205 //Pruebas de la funcion decodeFileToBase64
206
207 int test00DecodificoUsandoDosArchivosValidos() {
208   FILE* inputFile = fopen("textoCodificado.txt","r+");
209   FILE* outputFile = fopen("entrada.txt", "r+");
210
211   int resultado = decodeFileFromBase64(inputFile,outputFile);
212   char* inputFileDecoding = "En un lugar de La Mancha de cuyo nombre no quiero acordarme hola com
213
214   if (resultado == SUCCESS) {
215     char outputFileContent [MAX_LONGITUD];
216     memset(outputFileContent,0,MAX_LONGITUD);
217     fseek(outputFile, 0, SEEK_SET);
218     fread(outputFileContent,sizeof(char),MAX_LONGITUD,outputFile);
219
220     if (strcmp(inputFileDecoding,outputFileContent) == 0) {
221       printf("Codifico usando dos archivos validos : OK\n");
222       fclose(inputFile);
223       fclose(outputFile);
224       return SUCCESS;
225     } else {
226       printf("Codifico usando dos archivos validos : ERROR\n");
227       fclose(inputFile);
228       fclose(outputFile);
229       return EXIT_FAILURE;
230     }
231
232
233   } else {
234     printf("Codifico usando dos archivos validos : ERROR\n");
```

```
235     fclose(inputFile);
236     fclose(outputFile);
237     return EXIT_FAILURE;
238   }
239 }
240
241 int test01DecodificoUsandoArchivoDeInputInexistente() {
242   FILE* inputFile = fopen("noExiste.txt","r+");
243   FILE* outputFile = fopen("entrada.txt", "r+");
244
245   int resultado = decodeFileFromBase64(inputFile,outputFile);
246
247   if (resultado == EXIT_FAILURE) {
248     printf("Deodifico usando archivo de input inexistente : OK\n");
249     fclose(outputFile);
250     return SUCCESS;
251   } else {
252     printf("Decodifico usando archivo de input inexistente : ERROR\n");
253     fclose(inputFile);
254     fclose(outputFile);
255     return EXIT_FAILURE;
256   }
257 }
258
259 int test02DecodificoUsandoArchivoDeOutputInexistente() {
260   FILE* inputFile = fopen("textoCodificado.txt","r+");
261   FILE* outputFile = fopen("noExiste.txt", "r+");
262
263   int resultado = decodeFileFromBase64(inputFile,outputFile);
264
265   if (resultado == EXIT_FAILURE) {
266     printf("Decodifico usando archivo de output inexistente : OK\n");
267     fclose(inputFile);
268     return SUCCESS;
269   } else {
270     printf("Decodifico usando archivo de output inexistente : ERROR\n");
271     fclose(inputFile);
272     fclose(outputFile);
273     return EXIT_FAILURE;
274   }
275 }
276
277 int test03DecodificoUnArchivoBinario() {
278   FILE* inputFile = fopen("textoBinarioCodificado.bin","r+");
279   FILE* outputFile = fopen("pruebaBinario.bin", "r+");
280
281   int resultado = decodeFileFromBase64(inputFile,outputFile);
282   char* inputFileDecoding = "binario\n";
283
284   if (resultado == SUCCESS) {
285     char outputFileContent [MAX_LONGITUD];
286     memset(outputFileContent,0,MAX_LONGITUD);
287     fseek(outputFile, 0, SEEK_SET);
288     fread(outputFileContent,sizeof(char),MAX_LONGITUD,outputFile);
```

```
289
290
291     if (strcmp(inputFileDecoding,outputFileContent) == 0) {
292       printf("Decodifico un archivo binario: OK\n");
293       fclose(inputFile);
294       fclose(outputFile);
295       return SUCCESS;
296     } else {
297       printf("Decodifico un archivo binario : ERROR\n");
298       fclose(inputFile);
299       fclose(outputFile);
300       return EXIT_FAILURE;
301     }
302
303
304   } else {
305     printf("Codifico un archivo binario : ERROR\n");
306     fclose(inputFile);
307     fclose(outputFile);
308     return EXIT_FAILURE;
309   }
310 }
311
312
313 //Pruebas de uso
314
315 int test00PasoABase64UnaFraseYLuegoLaDecodifico() {
316   size_t outputLen = 0;
317   char* frase = "En un lugar de La Mancha de cuyo nombre no quiero acordarme";
318
319   char* encode = encodeBase64(frase,strlen(frase),&outputLen);
320   char* resultado = decodeBase64(encode,strlen(encode),&outputLen);
321
322   if (strcmp(frase,resultado) == 0) {
323     printf("Paso una frase a base64 y luego la decodifico : OK\n");
324     return SUCCESS;
325   } else {
326     printf("Paso una frase a base64 y luego la decodifico : ERROR\n");
327     return EXIT_FAILURE;
328   }
329 }
330
331
332
333 int tests() {
334   int resultado = 0;
335   printf("---Incio pruebas de la funcion encodeBase64---\n\n");
336   resultado += test00PasoABase64UnaFrase();
337   resultado += test01PasoABase64UnaFraseVacia();
338   resultado += test02PasoABase64UnaFraseInvalida();
339
340   printf("\n---Incio pruebas de la funcion decodeBase64---\n\n");
341   resultado += test00DecodificoUnaFraseEnBase64();
342   resultado += test01DecodificoUnaFraseVacia();
```

```
343   resultado += test02DecodificoUnaFraseInvalida();
344
345   printf("\n---Incio pruebas de la funcion encodeFileToBase64---\n\n");
346   resultado += test00CodificoUsandoDosArchivosValidos();
347   resultado += test01CodificoUsandoArchivoDeInputInexistente();
348   resultado += test02CodificoUsandoArchivoDeOutputInexistente();
349   resultado += test03CodificoUnArchivoBinario();
350
351   printf("\n---Incio pruebas de la funcion decodeFileToBase64---\n\n");
352   resultado += test00DecodificoUsandoDosArchivosValidos();
353   resultado += test01DecodificoUsandoArchivoDeInputInexistente();
354   resultado += test02DecodificoUsandoArchivoDeOutputInexistente();
355   resultado += test03DecodificoUnArchivoBinario();
356
357   printf("\n---Incio pruebas de uso---\n\n");
358   resultado += test00PasoABase64UnaFraseYLuegoLaDecodifico();
359
360   if (resultado == SUCCESS) return SUCCESS;
361   else return EXIT_FAILURE;
362 }
363
364 int main() {
365   return tests();
366 }
```

7.2. Código MIPS32, generado por el compilador

códigoMips32.c

```
1  tp0:     file format elf32-tradbigmips
2
3
4  Disassembly of section .init:
5
6  000008a0 <_init>:
7   8a0:  3c1c0002   lui  gp,0x2
8   8a4:  279c97f0   addiu  gp,gp,-26640
9   8a8:  0399e021   addu  gp,gp,t9
10  8ac:  27bdffe0   addiu  sp,sp,-32
11  8b0:  afbc0010   sw  gp,16(sp)
12  8b4:  afbf001c   sw  ra,28(sp)
13  8b8:  8f828094   lw  v0,-32620(gp)
14  8bc:  10400004   beqz  v0,8d0 <_init+0x30>
15  8c0:  00000000   nop
16  8c4:  8f998094   lw  t9,-32620(gp)
17  8c8:  0320f809   jalr  t9
18  8cc:  00000000   nop
19  8d0:  8fbf001c   lw  ra,28(sp)
20  8d4:  03e00008   jr  ra
21  8d8:  27bd0020   addiu  sp,sp,32
22
23 Disassembly of section .text:
24
25 000008e0 <__start>:
```

```
      8e0:  03e00025   move  zero,ra
      8e4:  04110001   bal  8ec <__start+0xc>
      8e8:  00000000   nop
      8ec:  3c1c0002   lui  gp,0x2
      8f0:  279c97a4   addiu  gp,gp,-26716
      8f4:  039fe021   addu  gp,gp,ra
      8f8:  0000f825   move  ra,zero
      8fc:  8f848018   lw  a0,-32744(gp)
      900:  8fa50000   lw  a1,0(sp)
      904:  27a60004   addiu  a2,sp,4
      908:  2401fff8   li  at,-8
      90c:  03a1e824   and  sp,sp,at
      910:  27bdffe0   addiu  sp,sp,-32
      914:  8f87801c   lw  a3,-32740(gp)
      918:  8f888020   lw  t0,-32736(gp)
      91c:  afa80010   sw  t0,16(sp)
      920:  afa20014   sw  v0,20(sp)
      924:  afbd0018   sw  sp,24(sp)
      928:  8f998088   lw  t9,-32632(gp)
      92c:  0320f809   jalr  t9
      930:  00000000   nop

00000934 <hlt>:
      934:  1000ffff   b  934 <hlt>
      938:  00000000   nop
      93c:  00000000   nop

00000940 <deregister_tm_clones>:
      940:  3c1c0002   lui  gp,0x2
      944:  279c9750   addiu  gp,gp,-26800
      948:  0399e021   addu  gp,gp,t9
      94c:  8f848028   lw  a0,-32728(gp)
      950:  8f828024   lw  v0,-32732(gp)
      954:  24842094   addiu  a0,a0,8340
      958:  24420003   addiu  v0,v0,3
      95c:  00441023   subu  v0,v0,a0
      960:  2c420007   sltiu  v0,v0,7
      964:  14400005   bnez  v0,97c <deregister_tm_clones+0x3c>
      968:  8f9980b0   lw  t9,-32592(gp)
      96c:  13200003   beqz  t9,97c <deregister_tm_clones+0x3c>
      970:  00000000   nop
      974:  03200008   jr  t9
      978:  00000000   nop
      97c:  03e00008   jr  ra
      980:  00000000   nop

00000984 <register_tm_clones>:
      984:  3c1c0002   lui  gp,0x2
      988:  279c970c   addiu  gp,gp,-26868
      98c:  0399e021   addu  gp,gp,t9
      990:  8f848028   lw  a0,-32728(gp)
      994:  8f858024   lw  a1,-32732(gp)
      998:  24842094   addiu  a0,a0,8340
      99c:  00a42823   subu  a1,a1,a0
```

```
80      9a0:  00052883   sra  a1,a1,0x2
81      9a4:  000517c2   srl  v0,a1,0x1f
82      9a8:  00452821   addu  a1,v0,a1
83      9ac:  00052843   sra  a1,a1,0x1
84      9b0:  10a00005   beqz  a1,9c8 <register_tm_clones+0x44>
85      9b4:  8f99806c   lw  t9,-32660(gp)
86      9b8:  13200003   beqz  t9,9c8 <register_tm_clones+0x44>
87      9bc:  00000000   nop
88      9c0:  03200008   jr  t9
89      9c4:  00000000   nop
90      9c8:  03e00008   jr  ra
91      9cc:  00000000   nop
92
93  000009d0 <__do_global_dtors_aux>:
94      9d0:  3c1c0002   lui  gp,0x2
95      9d4:  279c96c0   addiu  gp,gp,-26944
96      9d8:  0399e021   addu  gp,gp,t9
97      9dc:  27bdffe0   addiu  sp,sp,-32
98      9e0:  afb00018   sw  s0,24(sp)
99      9e4:  8f908028   lw  s0,-32728(gp)
100     9e8:  afbc0010   sw  gp,16(sp)
101     9ec:  afbf001c   sw  ra,28(sp)
102     9f0:  92022150   lbu  v0,8528(s0)
103     9f4:  1440000d   bnez  v0,a2c <__do_global_dtors_aux+0x5c>
104     9f8:  8f8280b4   lw  v0,-32588(gp)
105     9fc:  10400005   beqz  v0,a14 <__do_global_dtors_aux+0x44>
106     a00:  8f82802c   lw  v0,-32724(gp)
107     a04:  8f9980b4   lw  t9,-32588(gp)
108     a08:  0320f809   jalr  t9
109     a0c:  8c440000   lw  a0,0(v0)
110     a10:  8fbc0010   lw  gp,16(sp)
111     a14:  8f998030   lw  t9,-32720(gp)
112     a18:  27390940   addiu  t9,t9,2368
113     a1c:  0411ffc8   bal  940 <deregister_tm_clones>
114     a20:  00000000   nop
115     a24:  24020001   li  v0,1
116     a28:  a2022150   sb  v0,8528(s0)
117     a2c:  8fbf001c   lw  ra,28(sp)
118     a30:  8fb00018   lw  s0,24(sp)
119     a34:  03e00008   jr  ra
120     a38:  27bd0020   addiu  sp,sp,32
121
122 00000a3c <frame_dummy>:
123     a3c:  3c1c0002   lui  gp,0x2
124     a40:  279c9654   addiu  gp,gp,-27052
125     a44:  0399e021   addu  gp,gp,t9
126     a48:  8f828028   lw  v0,-32728(gp)
127     a4c:  27bdffe0   addiu  sp,sp,-32
128     a50:  2444201c   addiu  a0,v0,8220
129     a54:  afbc0010   sw  gp,16(sp)
130     a58:  afbf001c   sw  ra,28(sp)
131     a5c:  8c820000   lw  v0,0(a0)
132     a60:  14400006   bnez  v0,a7c <frame_dummy+0x40>
133     a64:  8f998074   lw  t9,-32652(gp)
```

```
134      a68:  8f998030   lw  t9,-32720(gp)
135      a6c:  8fbf001c   lw  ra,28(sp)
136      a70:  27390984   addiu  t9,t9,2436
137      a74:  1000ffc3   b  984 <register_tm_clones>
138      a78:  27bd0020   addiu  sp,sp,32
139      a7c:  1320fffa   beqz  t9,a68 <frame_dummy+0x2c>
140      a80:  00000000   nop
141      a84:  0320f809   jalr  t9
142      a88:  00000000   nop
143      a8c:  1000fff6   b  a68 <frame_dummy+0x2c>
144      a90:  8fbc0010   lw  gp,16(sp)
145    ...
146
147 00000aa0 <__b64_isvalidchar>:
148      aa0:  27bdffe8   addiu  sp,sp,-24
149      aa4:  afbe0014   sw  s8,20(sp)
150      aa8:  03a0f025   move  s8,sp
151      aac:  00801025   move  v0,a0
152      ab0:  a3c20018   sb  v0,24(s8)
153      ab4:  a3c00008   sb  zero,8(s8)
154      ab8:  83c20018   lb  v0,24(s8)
155      abc:  28420030   slti  v0,v0,48
156      ac0:  14400007   bnez  v0,ae0 <__b64_isvalidchar+0x40>
157      ac4:  00000000   nop
158      ac8:  83c20018   lb  v0,24(s8)
159      acc:  2842003a   slti  v0,v0,58
160      ad0:  10400003   beqz  v0,ae0 <__b64_isvalidchar+0x40>
161      ad4:  00000000   nop
162      ad8:  24020001   li  v0,1
163      adc:  a3c20008   sb  v0,8(s8)
164      ae0:  83c20018   lb  v0,24(s8)
165      ae4:  28420041   slti  v0,v0,65
166      ae8:  14400007   bnez  v0,b08 <__b64_isvalidchar+0x68>
167      aec:  00000000   nop
168      af0:  83c20018   lb  v0,24(s8)
169      af4:  2842005b   slti  v0,v0,91
170      af8:  10400003   beqz  v0,b08 <__b64_isvalidchar+0x68>
171      afc:  00000000   nop
172      b00:  24020001   li  v0,1
173      b04:  a3c20008   sb  v0,8(s8)
174      b08:  83c20018   lb  v0,24(s8)
175      b0c:  28420061   slti  v0,v0,97
176      b10:  14400007   bnez  v0,b30 <__b64_isvalidchar+0x90>
177      b14:  00000000   nop
178      b18:  83c20018   lb  v0,24(s8)
179      b1c:  2842007b   slti  v0,v0,123
180      b20:  10400003   beqz  v0,b30 <__b64_isvalidchar+0x90>
181      b24:  00000000   nop
182      b28:  24020001   li  v0,1
183      b2c:  a3c20008   sb  v0,8(s8)
184      b30:  83c30018   lb  v1,24(s8)
185      b34:  2402002b   li  v0,43
186      b38:  10620009   beq  v1,v0,b60 <__b64_isvalidchar+0xc0>
187      b3c:  00000000   nop
```

```
188      b40:  83c30018   lb  v1,24(s8)
189      b44:  2402002f   li  v0,47
190      b48:  10620005   beq  v1,v0,b60 <__b64_isvalidchar+0xc0>
191      b4c:  00000000   nop
192      b50:  83c30018   lb  v1,24(s8)
193      b54:  2402003d   li  v0,61
194      b58:  14620003   bne  v1,v0,b68 <__b64_isvalidchar+0xc8>
195      b5c:  00000000   nop
196      b60:  24020001   li  v0,1
197      b64:  a3c20008   sb  v0,8(s8)
198      b68:  93c20008   lbu  v0,8(s8)
199      b6c:  03c0e825   move  sp,s8
200      b70:  8fbe0014   lw  s8,20(sp)
201      b74:  27bd0018   addiu  sp,sp,24
202      b78:  03e00008   jr  ra
203      b7c:  00000000   nop
204
205 00000b80 <__len_base64_decode_output>:
206      b80:  27bdffe8   addiu  sp,sp,-24
207      b84:  afbe0014   sw  s8,20(sp)
208      b88:  03a0f025   move  s8,sp
209      b8c:  afc40018   sw  a0,24(s8)
210      b90:  afc5001c   sw  a1,28(s8)
211      b94:  8fc20018   lw  v0,24(s8)
212      b98:  14400004   bnez  v0,bac <__len_base64_decode_output+0x2c>
213      b9c:  00000000   nop
214      ba0:  00001025   move  v0,zero
215      ba4:  1000001e   b  c20 <__len_base64_decode_output+0xa0>
216      ba8:  00000000   nop
217      bac:  afc00008   sw  zero,8(s8)
218      bb0:  8fc2001c   lw  v0,28(s8)
219      bb4:  00021882   srl  v1,v0,0x2
220      bb8:  00601025   move  v0,v1
221      bbc:  00021040   sll  v0,v0,0x1
222      bc0:  00431021   addu  v0,v0,v1
223      bc4:  afc20008   sw  v0,8(s8)
224      bc8:  afc0000c   sw  zero,12(s8)
225      bcc:  1000000e   b  c08 <__len_base64_decode_output+0x88>
226      bd0:  00000000   nop
227      bd4:  8fc30018   lw  v1,24(s8)
228      bd8:  8fc2000c   lw  v0,12(s8)
229      bdc:  00621021   addu  v0,v1,v0
230      be0:  80430000   lb  v1,0(v0)
231      be4:  2402003d   li  v0,61
232      be8:  14620004   bne  v1,v0,bfc <__len_base64_decode_output+0x7c>
233      bec:  00000000   nop
234      bf0:  8fc20008   lw  v0,8(s8)
235      bf4:  2442ffff   addiu  v0,v0,-1
236      bf8:  afc20008   sw  v0,8(s8)
237      bfc:  8fc2000c   lw  v0,12(s8)
238      c00:  24420001   addiu  v0,v0,1
239      c04:  afc2000c   sw  v0,12(s8)
240      c08:  8fc3000c   lw  v1,12(s8)
241      c0c:  8fc2001c   lw  v0,28(s8)
```

```
242      c10:  0062102b   sltu  v0,v1,v0
243      c14:  1440ffef   bnez  v0,bd4 <__len_base64_decode_output+0x54>
244      c18:  00000000   nop
245      c1c:  8fc20008   lw  v0,8(s8)
246      c20:  03c0e825   move  sp,s8
247      c24:  8fbe0014   lw  s8,20(sp)
248      c28:  27bd0018   addiu  sp,sp,24
249      c2c:  03e00008   jr  ra
250      c30:  00000000   nop
251
252 00000c34 <__len_base64_encode_output>:
253      c34:  27bdffe8   addiu  sp,sp,-24
254      c38:  afbe0014   sw  s8,20(sp)
255      c3c:  03a0f025   move  s8,sp
256      c40:  afc40018   sw  a0,24(s8)
257      c44:  8fc20018   lw  v0,24(s8)
258      c48:  afc20008   sw  v0,8(s8)
259      c4c:  8fc40018   lw  a0,24(s8)
260      c50:  3c02aaaa   lui  v0,0xaaaa
261      c54:  3442aaab   ori  v0,v0,0xaaab
262      c58:  00820019   multu  a0,v0
263      c5c:  00001010   mfhi  v0
264      c60:  00021842   srl  v1,v0,0x1
265      c64:  00601025   move  v0,v1
266      c68:  00021040   sll  v0,v0,0x1
267      c6c:  00431021   addu  v0,v0,v1
268      c70:  00821823   subu  v1,a0,v0
269      c74:  1060000f   beqz  v1,cb4 <__len_base64_encode_output+0x80>
270      c78:  00000000   nop
271      c7c:  8fc40018   lw  a0,24(s8)
272      c80:  3c02aaaa   lui  v0,0xaaaa
273      c84:  3442aaab   ori  v0,v0,0xaaab
274      c88:  00820019   multu  a0,v0
275      c8c:  00001010   mfhi  v0
276      c90:  00021842   srl  v1,v0,0x1
277      c94:  00601025   move  v0,v1
278      c98:  00021040   sll  v0,v0,0x1
279      c9c:  00431021   addu  v0,v0,v1
280      ca0:  00821823   subu  v1,a0,v0
281      ca4:  8fc20008   lw  v0,8(s8)
282      ca8:  00431023   subu  v0,v0,v1
283      cac:  24420003   addiu  v0,v0,3
284      cb0:  afc20008   sw  v0,8(s8)
285      cb4:  8fc30008   lw  v1,8(s8)
286      cb8:  3c02aaaa   lui  v0,0xaaaa
287      cbc:  3442aaab   ori  v0,v0,0xaaab
288      cc0:  00620019   multu  v1,v0
289      cc4:  00001010   mfhi  v0
290      cc8:  00021042   srl  v0,v0,0x1
291      ccc:  afc20008   sw  v0,8(s8)
292      cd0:  8fc20008   lw  v0,8(s8)
293      cd4:  00021080   sll  v0,v0,0x2
294      cd8:  afc20008   sw  v0,8(s8)
295      cdc:  8fc20008   lw  v0,8(s8)
```

```
296      ce0:  03c0e825   move  sp,s8
297      ce4:  8fbe0014   lw  s8,20(sp)
298      ce8:  27bd0018   addiu  sp,sp,24
299      cec:  03e00008   jr  ra
300      cf0:  00000000   nop
301
302 00000cf4 <__write>:
303      cf4:  3c1c0002   lui  gp,0x2
304      cf8:  279c939c   addiu  gp,gp,-27748
305      cfc:  0399e021   addu  gp,gp,t9
306      d00:  27bdffe0   addiu  sp,sp,-32
307      d04:  afbf001c   sw  ra,28(sp)
308      d08:  afbe0018   sw  s8,24(sp)
309      d0c:  03a0f025   move  s8,sp
310      d10:  afbc0010   sw  gp,16(sp)
311      d14:  afc40020   sw  a0,32(s8)
312      d18:  afc50024   sw  a1,36(s8)
313      d1c:  afc60028   sw  a2,40(s8)
314      d20:  8fc20020   lw  v0,32(s8)
315      d24:  10400004   beqz  v0,d38 <__write+0x44>
316      d28:  00000000   nop
317      d2c:  8fc20024   lw  v0,36(s8)
318      d30:  14400004   bnez  v0,d44 <__write+0x50>
319      d34:  00000000   nop
320      d38:  24020001   li  v0,1
321      d3c:  1000000b   b  d6c <__write+0x78>
322      d40:  00000000   nop
323      d44:  8fc70020   lw  a3,32(s8)
324      d48:  8fc60028   lw  a2,40(s8)
325      d4c:  24050001   li  a1,1
326      d50:  8fc40024   lw  a0,36(s8)
327      d54:  8f82809c   lw  v0,-32612(gp)
328      d58:  0040c825   move  t9,v0
329      d5c:  0320f809   jalr  t9
330      d60:  00000000   nop
331      d64:  8fdc0010   lw  gp,16(s8)
332      d68:  00001025   move  v0,zero
333      d6c:  03c0e825   move  sp,s8
334      d70:  8fbf001c   lw  ra,28(sp)
335      d74:  8fbe0018   lw  s8,24(sp)
336      d78:  27bd0020   addiu  sp,sp,32
337      d7c:  03e00008   jr  ra
338      d80:  00000000   nop
339
340 00000d84 <__processFile>:
341      d84:  3c1c0002   lui  gp,0x2
342      d88:  279c930c   addiu  gp,gp,-27892
343      d8c:  0399e021   addu  gp,gp,t9
344      d90:  27bdffd0   addiu  sp,sp,-48
345      d94:  afbf002c   sw  ra,44(sp)
346      d98:  afbe0028   sw  s8,40(sp)
347      d9c:  03a0f025   move  s8,sp
348      da0:  afbc0010   sw  gp,16(sp)
349      da4:  afc40030   sw  a0,48(s8)
```

```
350     da8:  afc50034   sw  a1,52(s8)
351     dac:  afc60038   sw  a2,56(s8)
352     db0:  afc7003c   sw  a3,60(s8)
353     db4:  afc00018   sw  zero,24(s8)
354     db8:  afc00020   sw  zero,32(s8)
355     dbc:  afc0001c   sw  zero,28(s8)
356     dc0:  10000035   b  e98 <__processFile+0x114>
357     dc4:  00000000   nop
358     dc8:  93c20043   lbu  v0,67(s8)
359     dcc:  1040000d   beqz  v0,e04 <__processFile+0x80>
360     dd0:  00000000   nop
361     dd4:  27c20020   addiu  v0,s8,32
362     dd8:  00403025   move  a2,v0
363     ddc:  8fc5001c   lw  a1,28(s8)
364     de0:  8fc40038   lw  a0,56(s8)
365     de4:  8f828034   lw  v0,-32716(gp)
366     de8:  0040c825   move  t9,v0
367     dec:  04110045   bal  f04 <encodeBase64>
368     df0:  00000000   nop
369     df4:  8fdc0010   lw  gp,16(s8)
370     df8:  afc20018   sw  v0,24(s8)
371     dfc:  1000000b   b  e2c <__processFile+0xa8>
372     e00:  00000000   nop
373     e04:  27c20020   addiu  v0,s8,32
374     e08:  00403025   move  a2,v0
375     e0c:  8fc5001c   lw  a1,28(s8)
376     e10:  8fc40038   lw  a0,56(s8)
377     e14:  8f828038   lw  v0,-32712(gp)
378     e18:  0040c825   move  t9,v0
379     e1c:  04110107   bal  123c <decodeBase64>
380     e20:  00000000   nop
381     e24:  8fdc0010   lw  gp,16(s8)
382     e28:  afc20018   sw  v0,24(s8)
383     e2c:  8fc20018   lw  v0,24(s8)
384     e30:  10400011   beqz  v0,e78 <__processFile+0xf4>
385     e34:  00000000   nop
386     e38:  8fc20020   lw  v0,32(s8)
387     e3c:  00403025   move  a2,v0
388     e40:  8fc50018   lw  a1,24(s8)
389     e44:  8fc40034   lw  a0,52(s8)
390     e48:  8f828030   lw  v0,-32720(gp)
391     e4c:  24420cf4   addiu  v0,v0,3316
392     e50:  0040c825   move  t9,v0
393     e54:  0411ffa7   bal  cf4 <__write>
394     e58:  00000000   nop
395     e5c:  8fdc0010   lw  gp,16(s8)
396     e60:  8fc40018   lw  a0,24(s8)
397     e64:  8f8280a4   lw  v0,-32604(gp)
398     e68:  0040c825   move  t9,v0
399     e6c:  0320f809   jalr  t9
400     e70:  00000000   nop
401     e74:  8fdc0010   lw  gp,16(s8)
402     e78:  8fc6003c   lw  a2,60(s8)
403     e7c:  00002825   move  a1,zero
```

```
404      e80:  8fc40038   lw  a0,56(s8)
405      e84:  8f82807c   lw  v0,-32644(gp)
406      e88:  0040c825   move  t9,v0
407      e8c:  0320f809   jalr  t9
408      e90:  00000000   nop
409      e94:  8fdc0010   lw  gp,16(s8)
410      e98:  8fc70030   lw  a3,48(s8)
411      e9c:  8fc6003c   lw  a2,60(s8)
412      ea0:  24050001   li  a1,1
413      ea4:  8fc40038   lw  a0,56(s8)
414      ea8:  8f828098   lw  v0,-32616(gp)
415      eac:  0040c825   move  t9,v0
416      eb0:  0320f809   jalr  t9
417      eb4:  00000000   nop
418      eb8:  8fdc0010   lw  gp,16(s8)
419      ebc:  afc2001c   sw  v0,28(s8)
420      ec0:  8fc2001c   lw  v0,28(s8)
421      ec4:  1440ffc0   bnez  v0,dc8 <__processFile+0x44>
422      ec8:  00000000   nop
423      ecc:  8fc3001c   lw  v1,28(s8)
424      ed0:  2402ffff   li  v0,-1
425      ed4:  14620004   bne  v1,v0,ee8 <__processFile+0x164>
426      ed8:  00000000   nop
427      edc:  24020001   li  v0,1
428      ee0:  10000002   b  eec <__processFile+0x168>
429      ee4:  00000000   nop
430      ee8:  00001025   move  v0,zero
431      eec:  03c0e825   move  sp,s8
432      ef0:  8fbf002c   lw  ra,44(sp)
433      ef4:  8fbe0028   lw  s8,40(sp)
434      ef8:  27bd0030   addiu  sp,sp,48
435      efc:  03e00008   jr  ra
436      f00:  00000000   nop
437
438 00000f04 <encodeBase64>:
439      f04:  3c1c0002   lui  gp,0x2
440      f08:  279c918c   addiu  gp,gp,-28276
441      f0c:  0399e021   addu  gp,gp,t9
442      f10:  27bdffa8   addiu  sp,sp,-88
443      f14:  afbf0054   sw  ra,84(sp)
444      f18:  afbe0050   sw  s8,80(sp)
445      f1c:  03a0f025   move  s8,sp
446      f20:  afbc0010   sw  gp,16(sp)
447      f24:  afc40058   sw  a0,88(s8)
448      f28:  afc5005c   sw  a1,92(s8)
449      f2c:  afc60060   sw  a2,96(s8)
450      f30:  8fc20058   lw  v0,88(s8)
451      f34:  14400004   bnez  v0,f48 <encodeBase64+0x44>
452      f38:  00000000   nop
453      f3c:  00001025   move  v0,zero
454      f40:  100000b8   b  1224 <encodeBase64+0x320>
455      f44:  00000000   nop
456      f48:  8fc4005c   lw  a0,92(s8)
457      f4c:  8f828030   lw  v0,-32720(gp)
```

```
458     f50:  24420c34   addiu  v0,v0,3124
459     f54:  0040c825   move  t9,v0
460     f58:  0411ff36   bal  c34 <__len_base64_encode_output>
461     f5c:  00000000   nop
462     f60:  8fdc0010   lw  gp,16(s8)
463     f64:  afc20024   sw  v0,36(s8)
464     f68:  8fc20024   lw  v0,36(s8)
465     f6c:  24420001   addiu  v0,v0,1
466     f70:  00402825   move  a1,v0
467     f74:  24040001   li  a0,1
468     f78:  8f828064   lw  v0,-32668(gp)
469     f7c:  0040c825   move  t9,v0
470     f80:  0320f809   jalr  t9
471     f84:  00000000   nop
472     f88:  8fdc0010   lw  gp,16(s8)
473     f8c:  afc20028   sw  v0,40(s8)
474     f90:  8fc20028   lw  v0,40(s8)
475     f94:  14400004   bnez  v0,fa8 <encodeBase64+0xa4>
476     f98:  00000000   nop
477     f9c:  00001025   move  v0,zero
478     fa0:  100000a0   b  1224 <encodeBase64+0x320>
479     fa4:  00000000   nop
480     fa8:  afc00018   sw  zero,24(s8)
481     fac:  afc0001c   sw  zero,28(s8)
482     fb0:  1000006c   b  1164 <encodeBase64+0x260>
483     fb4:  00000000   nop
484     fb8:  8fc20018   lw  v0,24(s8)
485     fbc:  24430001   addiu  v1,v0,1
486     fc0:  afc30018   sw  v1,24(s8)
487     fc4:  8fc30058   lw  v1,88(s8)
488     fc8:  00621021   addu  v0,v1,v0
489     fcc:  80420000   lb  v0,0(v0)
490     fd0:  304200ff   andi  v0,v0,0xff
491     fd4:  afc2002c   sw  v0,44(s8)
492     fd8:  8fc30018   lw  v1,24(s8)
493     fdc:  8fc2005c   lw  v0,92(s8)
494     fe0:  0062102b   sltu  v0,v1,v0
495     fe4:  10400009   beqz  v0,100c <encodeBase64+0x108>
496     fe8:  00000000   nop
497     fec:  8fc20018   lw  v0,24(s8)
498     ff0:  24430001   addiu  v1,v0,1
499     ff4:  afc30018   sw  v1,24(s8)
500     ff8:  8fc30058   lw  v1,88(s8)
501     ffc:  00621021   addu  v0,v1,v0
502    1000:  80420000   lb  v0,0(v0)
503    1004:  10000002   b  1010 <encodeBase64+0x10c>
504    1008:  00000000   nop
505    100c:  00001025   move  v0,zero
506    1010:  afc20030   sw  v0,48(s8)
507    1014:  8fc30018   lw  v1,24(s8)
508    1018:  8fc2005c   lw  v0,92(s8)
509    101c:  0062102b   sltu  v0,v1,v0
510    1020:  10400009   beqz  v0,1048 <encodeBase64+0x144>
511    1024:  00000000   nop
```

```
512     1028:  8fc20018   lw  v0,24(s8)
513     102c:  24430001   addiu  v1,v0,1
514     1030:  afc30018   sw  v1,24(s8)
515     1034:  8fc30058   lw  v1,88(s8)
516     1038:  00621021   addu  v0,v1,v0
517     103c:  80420000   lb  v0,0(v0)
518     1040:  10000002   b  104c <encodeBase64+0x148>
519     1044:  00000000   nop
520     1048:  00001025   move  v0,zero
521     104c:  afc20034   sw  v0,52(s8)
522     1050:  8fc2002c   lw  v0,44(s8)
523     1054:  00021c00   sll  v1,v0,0x10
524     1058:  8fc20030   lw  v0,48(s8)
525     105c:  00021200   sll  v0,v0,0x8
526     1060:  00621821   addu  v1,v1,v0
527     1064:  8fc20034   lw  v0,52(s8)
528     1068:  00621021   addu  v0,v1,v0
529     106c:  afc20038   sw  v0,56(s8)
530     1070:  8fc20038   lw  v0,56(s8)
531     1074:  00021482   srl  v0,v0,0x12
532     1078:  3042003f   andi  v0,v0,0x3f
533     107c:  afc2003c   sw  v0,60(s8)
534     1080:  8fc20038   lw  v0,56(s8)
535     1084:  00021302   srl  v0,v0,0xc
536     1088:  3042003f   andi  v0,v0,0x3f
537     108c:  afc20040   sw  v0,64(s8)
538     1090:  8fc20038   lw  v0,56(s8)
539     1094:  00021182   srl  v0,v0,0x6
540     1098:  3042003f   andi  v0,v0,0x3f
541     109c:  afc20044   sw  v0,68(s8)
542     10a0:  8fc20038   lw  v0,56(s8)
543     10a4:  3042003f   andi  v0,v0,0x3f
544     10a8:  afc20048   sw  v0,72(s8)
545     10ac:  8fc30028   lw  v1,40(s8)
546     10b0:  8fc2001c   lw  v0,28(s8)
547     10b4:  00621021   addu  v0,v1,v0
548     10b8:  8f838030   lw  v1,-32720(gp)
549     10bc:  24641df0   addiu  a0,v1,7664
550     10c0:  8fc3003c   lw  v1,60(s8)
551     10c4:  00831821   addu  v1,a0,v1
552     10c8:  90630000   lbu  v1,0(v1)
553     10cc:  7c031c20   seb  v1,v1
554     10d0:  a0430000   sb  v1,0(v0)
555     10d4:  8fc2001c   lw  v0,28(s8)
556     10d8:  24420001   addiu  v0,v0,1
557     10dc:  8fc30028   lw  v1,40(s8)
558     10e0:  00621021   addu  v0,v1,v0
559     10e4:  8f838030   lw  v1,-32720(gp)
560     10e8:  24641df0   addiu  a0,v1,7664
561     10ec:  8fc30040   lw  v1,64(s8)
562     10f0:  00831821   addu  v1,a0,v1
563     10f4:  90630000   lbu  v1,0(v1)
564     10f8:  7c031c20   seb  v1,v1
565     10fc:  a0430000   sb  v1,0(v0)
```

```
566     1100:  8fc2001c   lw  v0,28(s8)
567     1104:  24420002   addiu  v0,v0,2
568     1108:  8fc30028   lw  v1,40(s8)
569     110c:  00621021   addu  v0,v1,v0
570     1110:  8f838030   lw  v1,-32720(gp)
571     1114:  24641df0   addiu  a0,v1,7664
572     1118:  8fc30044   lw  v1,68(s8)
573     111c:  00831821   addu  v1,a0,v1
574     1120:  90630000   lbu  v1,0(v1)
575     1124:  7c031c20   seb  v1,v1
576     1128:  a0430000   sb  v1,0(v0)
577     112c:  8fc2001c   lw  v0,28(s8)
578     1130:  24420003   addiu  v0,v0,3
579     1134:  8fc30028   lw  v1,40(s8)
580     1138:  00621021   addu  v0,v1,v0
581     113c:  8f838030   lw  v1,-32720(gp)
582     1140:  24641df0   addiu  a0,v1,7664
583     1144:  8fc30048   lw  v1,72(s8)
584     1148:  00831821   addu  v1,a0,v1
585     114c:  90630000   lbu  v1,0(v1)
586     1150:  7c031c20   seb  v1,v1
587     1154:  a0430000   sb  v1,0(v0)
588     1158:  8fc2001c   lw  v0,28(s8)
589     115c:  24420004   addiu  v0,v0,4
590     1160:  afc2001c   sw  v0,28(s8)
591     1164:  8fc30018   lw  v1,24(s8)
592     1168:  8fc2005c   lw  v0,92(s8)
593     116c:  0062102b   sltu  v0,v1,v0
594     1170:  1440ff91   bnez  v0,fb8 <encodeBase64+0xb4>
595     1174:  00000000   nop
596     1178:  afc00020   sw  zero,32(s8)
597     117c:  1000000c   b  11b0 <encodeBase64+0x2ac>
598     1180:  00000000   nop
599     1184:  8fc20020   lw  v0,32(s8)
600     1188:  8fc30024   lw  v1,36(s8)
601     118c:  00621023   subu  v0,v1,v0
602     1190:  2442ffff   addiu  v0,v0,-1
603     1194:  8fc30028   lw  v1,40(s8)
604     1198:  00621021   addu  v0,v1,v0
605     119c:  2403003d   li  v1,61
606     11a0:  a0430000   sb  v1,0(v0)
607     11a4:  8fc20020   lw  v0,32(s8)
608     11a8:  24420001   addiu  v0,v0,1
609     11ac:  afc20020   sw  v0,32(s8)
610     11b0:  8fc4005c   lw  a0,92(s8)
611     11b4:  3c02aaaa   lui  v0,0xaaaa
612     11b8:  3442aaab   ori  v0,v0,0xaaab
613     11bc:  00820019   multu  a0,v0
614     11c0:  00001010   mfhi  v0
615     11c4:  00021842   srl  v1,v0,0x1
616     11c8:  00601025   move  v0,v1
617     11cc:  00021040   sll  v0,v0,0x1
618     11d0:  00431021   addu  v0,v0,v1
619     11d4:  00821823   subu  v1,a0,v0
```

```
620     11d8:  8f828028   lw  v0,-32728(gp)
621     11dc:  00031880   sll  v1,v1,0x2
622     11e0:  24422030   addiu  v0,v0,8240
623     11e4:  00621021   addu  v0,v1,v0
624     11e8:  8c430000   lw  v1,0(v0)
625     11ec:  8fc20020   lw  v0,32(s8)
626     11f0:  0043102a   slt  v0,v0,v1
627     11f4:  1440ffe3   bnez  v0,1184 <encodeBase64+0x280>
628     11f8:  00000000   nop
629     11fc:  8fc40028   lw  a0,40(s8)
630     1200:  8f82808c   lw  v0,-32628(gp)
631     1204:  0040c825   move  t9,v0
632     1208:  0320f809   jalr  t9
633     120c:  00000000   nop
634     1210:  8fdc0010   lw  gp,16(s8)
635     1214:  00401825   move  v1,v0
636     1218:  8fc20060   lw  v0,96(s8)
637     121c:  ac430000   sw  v1,0(v0)
638     1220:  8fc20028   lw  v0,40(s8)
639     1224:  03c0e825   move  sp,s8
640     1228:  8fbf0054   lw  ra,84(sp)
641     122c:  8fbe0050   lw  s8,80(sp)
642     1230:  27bd0058   addiu  sp,sp,88
643     1234:  03e00008   jr  ra
644     1238:  00000000   nop
645
646 0000123c <decodeBase64>:
647     123c:  3c1c0002   lui  gp,0x2
648     1240:  279c8e54   addiu  gp,gp,-29100
649     1244:  0399e021   addu  gp,gp,t9
650     1248:  27bdfec0   addiu  sp,sp,-320
651     124c:  afbf013c   sw  ra,316(sp)
652     1250:  afbe0138   sw  s8,312(sp)
653     1254:  03a0f025   move  s8,sp
654     1258:  afbc0010   sw  gp,16(sp)
655     125c:  afc40140   sw  a0,320(s8)
656     1260:  afc50144   sw  a1,324(s8)
657     1264:  afc60148   sw  a2,328(s8)
658     1268:  8fc20140   lw  v0,320(s8)
659     126c:  14400004   bnez  v0,1280 <decodeBase64+0x44>
660     1270:  00000000   nop
661     1274:  00001025   move  v0,zero
662     1278:  100000d3   b  15c8 <decodeBase64+0x38c>
663     127c:  00000000   nop
664     1280:  8fc20140   lw  v0,320(s8)
665     1284:  afc20024   sw  v0,36(s8)
666     1288:  27c20034   addiu  v0,s8,52
667     128c:  24060100   li  a2,256
668     1290:  2405ffff   li  a1,-1
669     1294:  00402025   move  a0,v0
670     1298:  8f82807c   lw  v0,-32644(gp)
671     129c:  0040c825   move  t9,v0
672     12a0:  0320f809   jalr  t9
673     12a4:  00000000   nop
```

```
12a8:  8fdc0010   lw  gp,16(s8)
12ac:  afc00018   sw  zero,24(s8)
12b0:  1000000f   b  12f0 <decodeBase64+0xb4>
12b4:  00000000   nop
12b8:  8f828030   lw  v0,-32720(gp)
12bc:  24431df0   addiu  v1,v0,7664
12c0:  8fc20018   lw  v0,24(s8)
12c4:  00621021   addu  v0,v1,v0
12c8:  90420000   lbu  v0,0(v0)
12cc:  00402025   move  a0,v0
12d0:  8fc20018   lw  v0,24(s8)
12d4:  304300ff   andi  v1,v0,0xff
12d8:  27c20018   addiu  v0,s8,24
12dc:  00441021   addu  v0,v0,a0
12e0:  a043001c   sb  v1,28(v0)
12e4:  8fc20018   lw  v0,24(s8)
12e8:  24420001   addiu  v0,v0,1
12ec:  afc20018   sw  v0,24(s8)
12f0:  8fc20018   lw  v0,24(s8)
12f4:  2c420040   sltiu  v0,v0,64
12f8:  1440ffef   bnez  v0,12b8 <decodeBase64+0x7c>
12fc:  00000000   nop
1300:  8fc50144   lw  a1,324(s8)
1304:  8fc40140   lw  a0,320(s8)
1308:  8f828030   lw  v0,-32720(gp)
130c:  24420b80   addiu  v0,v0,2944
1310:  0040c825   move  t9,v0
1314:  0411fe1a   bal  b80 <__len_base64_decode_output>
1318:  00000000   nop
131c:  8fdc0010   lw  gp,16(s8)
1320:  afc20028   sw  v0,40(s8)
1324:  8fc20028   lw  v0,40(s8)
1328:  24420001   addiu  v0,v0,1
132c:  00402825   move  a1,v0
1330:  24040001   li  a0,1
1334:  8f828064   lw  v0,-32668(gp)
1338:  0040c825   move  t9,v0
133c:  0320f809   jalr  t9
1340:  00000000   nop
1344:  8fdc0010   lw  gp,16(s8)
1348:  afc2002c   sw  v0,44(s8)
134c:  8fc2002c   lw  v0,44(s8)
1350:  14400004   bnez  v0,1364 <decodeBase64+0x128>
1354:  00000000   nop
1358:  00001025   move  v0,zero
135c:  1000009a   b  15c8 <decodeBase64+0x38c>
1360:  00000000   nop
1364:  afc0001c   sw  zero,28(s8)
1368:  afc00020   sw  zero,32(s8)
136c:  10000087   b  158c <decodeBase64+0x350>
1370:  00000000   nop
1374:  8fc30140   lw  v1,320(s8)
1378:  8fc2001c   lw  v0,28(s8)
137c:  00621021   addu  v0,v1,v0
```

```
728     1380:   80420000   lb  v0,0(v0)
729     1384:   00402025   move  a0,v0
730     1388:   8f828030   lw  v0,-32720(gp)
731     138c:   24420aa0   addiu  v0,v0,2720
732     1390:   0040c825   move  t9,v0
733     1394:   0411fdc2   bal  aa0 <__b64_isvalidchar>
734     1398:   00000000   nop
735     139c:   8fdc0010   lw  gp,16(s8)
736     13a0:   38420001   xori  v0,v0,0x1
737     13a4:   304200ff   andi  v0,v0,0xff
738     13a8:   14400071   bnez  v0,1570 <decodeBase64+0x334>
739     13ac:   00000000   nop
740     13b0:   8fc30024   lw  v1,36(s8)
741     13b4:   8fc2001c   lw  v0,28(s8)
742     13b8:   00621021   addu  v0,v1,v0
743     13bc:   90420000   lbu  v0,0(v0)
744     13c0:   00401825   move  v1,v0
745     13c4:   27c20018   addiu  v0,s8,24
746     13c8:   00431021   addu  v0,v0,v1
747     13cc:   9042001c   lbu  v0,28(v0)
748     13d0:   afc20030   sw  v0,48(s8)
749     13d4:   8fc20030   lw  v0,48(s8)
750     13d8:   00021180   sll  v0,v0,0x6
751     13dc:   8fc3001c   lw  v1,28(s8)
752     13e0:   24630001   addiu  v1,v1,1
753     13e4:   8fc40024   lw  a0,36(s8)
754     13e8:   00831821   addu  v1,a0,v1
755     13ec:   90630000   lbu  v1,0(v1)
756     13f0:   00602025   move  a0,v1
757     13f4:   27c30018   addiu  v1,s8,24
758     13f8:   00641821   addu  v1,v1,a0
759     13fc:   9063001c   lbu  v1,28(v1)
760     1400:   00431025   or  v0,v0,v1
761     1404:   afc20030   sw  v0,48(s8)
762     1408:   8fc2001c   lw  v0,28(s8)
763     140c:   24420002   addiu  v0,v0,2
764     1410:   8fc30140   lw  v1,320(s8)
765     1414:   00621021   addu  v0,v1,v0
766     1418:   80430000   lb  v1,0(v0)
767     141c:   2402003d   li  v0,61
768     1420:   14620005   bne  v1,v0,1438 <decodeBase64+0x1fc>
769     1424:   00000000   nop
770     1428:   8fc20030   lw  v0,48(s8)
771     142c:   00021180   sll  v0,v0,0x6
772     1430:   1000000d   b  1468 <decodeBase64+0x22c>
773     1434:   00000000   nop
774     1438:   8fc20030   lw  v0,48(s8)
775     143c:   00021180   sll  v0,v0,0x6
776     1440:   8fc3001c   lw  v1,28(s8)
777     1444:   24630002   addiu  v1,v1,2
778     1448:   8fc40024   lw  a0,36(s8)
779     144c:   00831821   addu  v1,a0,v1
780     1450:   90630000   lbu  v1,0(v1)
781     1454:   00602025   move  a0,v1
```

```
782     1458:   27c30018    addiu  v1,s8,24
783     145c:   00641821    addu  v1,v1,a0
784     1460:   9063001c    lbu  v1,28(v1)
785     1464:   00431025    or  v0,v0,v1
786     1468:   afc20030    sw  v0,48(s8)
787     146c:   8fc2001c    lw  v0,28(s8)
788     1470:   24420003    addiu  v0,v0,3
789     1474:   8fc30140    lw  v1,320(s8)
790     1478:   00621021    addu  v0,v1,v0
791     147c:   80430000    lb  v1,0(v0)
792     1480:   2402003d    li  v0,61
793     1484:   14620005    bne  v1,v0,149c <decodeBase64+0x260>
794     1488:   00000000    nop
795     148c:   8fc20030    lw  v0,48(s8)
796     1490:   00021180    sll  v0,v0,0x6
797     1494:   1000000d    b  14cc <decodeBase64+0x290>
798     1498:   00000000    nop
799     149c:   8fc20030    lw  v0,48(s8)
800     14a0:   00021180    sll  v0,v0,0x6
801     14a4:   8fc3001c    lw  v1,28(s8)
802     14a8:   24630003    addiu  v1,v1,3
803     14ac:   8fc40024    lw  a0,36(s8)
804     14b0:   00831821    addu  v1,a0,v1
805     14b4:   90630000    lbu  v1,0(v1)
806     14b8:   00602025    move  a0,v1
807     14bc:   27c30018    addiu  v1,s8,24
808     14c0:   00641821    addu  v1,v1,a0
809     14c4:   9063001c    lbu  v1,28(v1)
810     14c8:   00431025    or  v0,v0,v1
811     14cc:   afc20030    sw  v0,48(s8)
812     14d0:   8fc3002c    lw  v1,44(s8)
813     14d4:   8fc20020    lw  v0,32(s8)
814     14d8:   00621021    addu  v0,v1,v0
815     14dc:   8fc30030    lw  v1,48(s8)
816     14e0:   00031c02    srl  v1,v1,0x10
817     14e4:   7c031c20    seb  v1,v1
818     14e8:   a0430000    sb  v1,0(v0)
819     14ec:   8fc2001c    lw  v0,28(s8)
820     14f0:   24420002    addiu  v0,v0,2
821     14f4:   8fc30140    lw  v1,320(s8)
822     14f8:   00621021    addu  v0,v1,v0
823     14fc:   80430000    lb  v1,0(v0)
824     1500:   2402003d    li  v0,61
825     1504:   10620009    beq  v1,v0,152c <decodeBase64+0x2f0>
826     1508:   00000000    nop
827     150c:   8fc20020    lw  v0,32(s8)
828     1510:   24420001    addiu  v0,v0,1
829     1514:   8fc3002c    lw  v1,44(s8)
830     1518:   00621021    addu  v0,v1,v0
831     151c:   8fc30030    lw  v1,48(s8)
832     1520:   00031a02    srl  v1,v1,0x8
833     1524:   7c031c20    seb  v1,v1
834     1528:   a0430000    sb  v1,0(v0)
835     152c:   8fc2001c    lw  v0,28(s8)
```

```
    1530:  24420003   addiu  v0,v0,3
    1534:  8fc30140   lw  v1,320(s8)
    1538:  00621021   addu  v0,v1,v0
    153c:  80430000   lb  v1,0(v0)
    1540:  2402003d   li  v0,61
    1544:  1062000b   beq  v1,v0,1574 <decodeBase64+0x338>
    1548:  00000000   nop
    154c:  8fc20020   lw  v0,32(s8)
    1550:  24420002   addiu  v0,v0,2
    1554:  8fc3002c   lw  v1,44(s8)
    1558:  00621021   addu  v0,v1,v0
    155c:  8fc30030   lw  v1,48(s8)
    1560:  7c031c20   seb  v1,v1
    1564:  a0430000   sb  v1,0(v0)
    1568:  10000002   b  1574 <decodeBase64+0x338>
    156c:  00000000   nop
    1570:  00000000   nop
    1574:  8fc2001c   lw  v0,28(s8)
    1578:  24420004   addiu  v0,v0,4
    157c:  afc2001c   sw  v0,28(s8)
    1580:  8fc20020   lw  v0,32(s8)
    1584:  24420003   addiu  v0,v0,3
    1588:  afc20020   sw  v0,32(s8)
    158c:  8fc3001c   lw  v1,28(s8)
    1590:  8fc20144   lw  v0,324(s8)
    1594:  0062102b   sltu  v0,v1,v0
    1598:  1440ff76   bnez  v0,1374 <decodeBase64+0x138>
    159c:  00000000   nop
    15a0:  8fc4002c   lw  a0,44(s8)
    15a4:  8f82808c   lw  v0,-32628(gp)
    15a8:  0040c825   move  t9,v0
    15ac:  0320f809   jalr  t9
    15b0:  00000000   nop
    15b4:  8fdc0010   lw  gp,16(s8)
    15b8:  00401825   move  v1,v0
    15bc:  8fc20148   lw  v0,328(s8)
    15c0:  ac430000   sw  v1,0(v0)
    15c4:  8fc2002c   lw  v0,44(s8)
    15c8:  03c0e825   move  sp,s8
    15cc:  8fbf013c   lw  ra,316(sp)
    15d0:  8fbe0138   lw  s8,312(sp)
    15d4:  27bd0140   addiu  sp,sp,320
    15d8:  03e00008   jr  ra
    15dc:  00000000   nop

000015e0 <encodeFileToBase64>:
    15e0:  3c1c0002   lui  gp,0x2
    15e4:  279c8ab0   addiu  gp,gp,-30032
    15e8:  0399e021   addu  gp,gp,t9
    15ec:  27bdffd0   addiu  sp,sp,-48
    15f0:  afbf002c   sw  ra,44(sp)
    15f4:  afbe0028   sw  s8,40(sp)
    15f8:  03a0f025   move  s8,sp
    15fc:  afbc0018   sw  gp,24(sp)
```

```
    1600:  afc40030   sw  a0,48(s8)
    1604:  afc50034   sw  a1,52(s8)
    1608:  8fc20030   lw  v0,48(s8)
    160c:  10400004   beqz  v0,1620 <encodeFileToBase64+0x40>
    1610:  00000000   nop
    1614:  8fc20034   lw  v0,52(s8)
    1618:  14400004   bnez  v0,162c <encodeFileToBase64+0x4c>
    161c:  00000000   nop
    1620:  24020001   li  v0,1
    1624:  10000017   b  1684 <encodeFileToBase64+0xa4>
    1628:  00000000   nop
    162c:  24060003   li  a2,3
    1630:  00002825   move  a1,zero
    1634:  27c20020   addiu  v0,s8,32
    1638:  00402025   move  a0,v0
    163c:  8f82807c   lw  v0,-32644(gp)
    1640:  0040c825   move  t9,v0
    1644:  0320f809   jalr  t9
    1648:  00000000   nop
    164c:  8fdc0018   lw  gp,24(s8)
    1650:  24020001   li  v0,1
    1654:  afa20010   sw  v0,16(sp)
    1658:  24070003   li  a3,3
    165c:  27c20020   addiu  v0,s8,32
    1660:  00403025   move  a2,v0
    1664:  8fc50034   lw  a1,52(s8)
    1668:  8fc40030   lw  a0,48(s8)
    166c:  8f828030   lw  v0,-32720(gp)
    1670:  24420d84   addiu  v0,v0,3460
    1674:  0040c825   move  t9,v0
    1678:  0411fdc2   bal  d84 <__processFile>
    167c:  00000000   nop
    1680:  8fdc0018   lw  gp,24(s8)
    1684:  03c0e825   move  sp,s8
    1688:  8fbf002c   lw  ra,44(sp)
    168c:  8fbe0028   lw  s8,40(sp)
    1690:  27bd0030   addiu  sp,sp,48
    1694:  03e00008   jr  ra
    1698:  00000000   nop

0000169c <decodeFileFromBase64>:
    169c:  3c1c0002   lui  gp,0x2
    16a0:  279c89f4   addiu  gp,gp,-30220
    16a4:  0399e021   addu  gp,gp,t9
    16a8:  27bdffd0   addiu  sp,sp,-48
    16ac:  afbf002c   sw  ra,44(sp)
    16b0:  afbe0028   sw  s8,40(sp)
    16b4:  03a0f025   move  s8,sp
    16b8:  afbc0018   sw  gp,24(sp)
    16bc:  afc40030   sw  a0,48(s8)
    16c0:  afc50034   sw  a1,52(s8)
    16c4:  8fc20030   lw  v0,48(s8)
    16c8:  10400004   beqz  v0,16dc <decodeFileFromBase64+0x40>
    16cc:  00000000   nop
```

```
    16d0:  8fc20034   lw  v0,52(s8)
    16d4:  14400004   bnez  v0,16e8 <decodeFileFromBase64+0x4c>
    16d8:  00000000   nop
    16dc:  24020001   li  v0,1
    16e0:  10000016   b  173c <decodeFileFromBase64+0xa0>
    16e4:  00000000   nop
    16e8:  24060004   li  a2,4
    16ec:  00002825   move  a1,zero
    16f0:  27c20020   addiu  v0,s8,32
    16f4:  00402025   move  a0,v0
    16f8:  8f82807c   lw  v0,-32644(gp)
    16fc:  0040c825   move  t9,v0
    1700:  0320f809   jalr  t9
    1704:  00000000   nop
    1708:  8fdc0018   lw  gp,24(s8)
    170c:  afa00010   sw  zero,16(sp)
    1710:  24070004   li  a3,4
    1714:  27c20020   addiu  v0,s8,32
    1718:  00403025   move  a2,v0
    171c:  8fc50034   lw  a1,52(s8)
    1720:  8fc40030   lw  a0,48(s8)
    1724:  8f828030   lw  v0,-32720(gp)
    1728:  24420d84   addiu  v0,v0,3460
    172c:  0040c825   move  t9,v0
    1730:  0411fd94   bal  d84 <__processFile>
    1734:  00000000   nop
    1738:  8fdc0018   lw  gp,24(s8)
    173c:  03c0e825   move  sp,s8
    1740:  8fbf002c   lw  ra,44(sp)
    1744:  8fbe0028   lw  s8,40(sp)
    1748:  27bd0030   addiu  sp,sp,48
    174c:  03e00008   jr  ra
    1750:  00000000   nop
  ...

00001760 <imprimir_salida>:
    1760:  3c1c0002   lui  gp,0x2
    1764:  279c8930   addiu  gp,gp,-30416
    1768:  0399e021   addu  gp,gp,t9
    176c:  27bdffe0   addiu  sp,sp,-32
    1770:  afbf001c   sw  ra,28(sp)
    1774:  afbe0018   sw  s8,24(sp)
    1778:  03a0f025   move  s8,sp
    177c:  afbc0010   sw  gp,16(sp)
    1780:  afc40020   sw  a0,32(s8)
    1784:  8f828078   lw  v0,-32648(gp)
    1788:  8c430000   lw  v1,0(v0)
    178c:  8fc60020   lw  a2,32(s8)
    1790:  8f828030   lw  v0,-32720(gp)
    1794:  24451e40   addiu  a1,v0,7744
    1798:  00602025   move  a0,v1
    179c:  8f828084   lw  v0,-32636(gp)
    17a0:  0040c825   move  t9,v0
    17a4:  0320f809   jalr  t9
```

```
998      17a8:  00000000   nop
999      17ac:  8fdc0010   lw  gp,16(s8)
1000     17b0:  00000000   nop
1001     17b4:  03c0e825   move  sp,s8
1002     17b8:  8fbf001c   lw  ra,28(sp)
1003     17bc:  8fbe0018   lw  s8,24(sp)
1004     17c0:  27bd0020   addiu  sp,sp,32
1005     17c4:  03e00008   jr  ra
1006     17c8:  00000000   nop
1007
1008 000017cc <imprimir_error>:
1009     17cc:  3c1c0002   lui  gp,0x2
1010     17d0:  279c88c4   addiu  gp,gp,-30524
1011     17d4:  0399e021   addu  gp,gp,t9
1012     17d8:  27bdffe0   addiu  sp,sp,-32
1013     17dc:  afbf001c   sw  ra,28(sp)
1014     17e0:  afbe0018   sw  s8,24(sp)
1015     17e4:  03a0f025   move  s8,sp
1016     17e8:  afbc0010   sw  gp,16(sp)
1017     17ec:  afc40020   sw  a0,32(s8)
1018     17f0:  8f8280a8   lw  v0,-32600(gp)
1019     17f4:  8c430000   lw  v1,0(v0)
1020     17f8:  8fc60020   lw  a2,32(s8)
1021     17fc:  8f828030   lw  v0,-32720(gp)
1022     1800:  24451e40   addiu  a1,v0,7744
1023     1804:  00602025   move  a0,v1
1024     1808:  8f828084   lw  v0,-32636(gp)
1025     180c:  0040c825   move  t9,v0
1026     1810:  0320f809   jalr  t9
1027     1814:  00000000   nop
1028     1818:  8fdc0010   lw  gp,16(s8)
1029     181c:  00000000   nop
1030     1820:  03c0e825   move  sp,s8
1031     1824:  8fbf001c   lw  ra,28(sp)
1032     1828:  8fbe0018   lw  s8,24(sp)
1033     182c:  27bd0020   addiu  sp,sp,32
1034     1830:  03e00008   jr  ra
1035     1834:  00000000   nop
1036
1037 00001838 <main>:
1038     1838:  3c1c0002   lui  gp,0x2
1039     183c:  279c8858   addiu  gp,gp,-30632
1040     1840:  0399e021   addu  gp,gp,t9
1041     1844:  27bdfdb0   addiu  sp,sp,-592
1042     1848:  afbf024c   sw  ra,588(sp)
1043     184c:  afbe0248   sw  s8,584(sp)
1044     1850:  afb00244   sw  s0,580(sp)
1045     1854:  03a0f025   move  s8,sp
1046     1858:  afbc0018   sw  gp,24(sp)
1047     185c:  afc40250   sw  a0,592(s8)
1048     1860:  afc50254   sw  a1,596(s8)
1049     1864:  24020001   li  v0,1
1050     1868:  afc20020   sw  v0,32(s8)
1051     186c:  afc00024   sw  zero,36(s8)
```

```
1870:  afc00028   sw  zero,40(s8)
1874:  27c20038   addiu  v0,s8,56
1878:  24060100   li  a2,256
187c:  00002825   move  a1,zero
1880:  00402025   move  a0,v0
1884:  8f82807c   lw  v0,-32644(gp)
1888:  0040c825   move  t9,v0
188c:  0320f809   jalr  t9
1890:  00000000   nop
1894:  8fdc0018   lw  gp,24(s8)
1898:  27c20138   addiu  v0,s8,312
189c:  24060100   li  a2,256
18a0:  00002825   move  a1,zero
18a4:  00402025   move  a0,v0
18a8:  8f82807c   lw  v0,-32644(gp)
18ac:  0040c825   move  t9,v0
18b0:  0320f809   jalr  t9
18b4:  00000000   nop
18b8:  8fdc0018   lw  gp,24(s8)
18bc:  afc00238   sw  zero,568(s8)
18c0:  27c20238   addiu  v0,s8,568
18c4:  afa20010   sw  v0,16(sp)
18c8:  8f828028   lw  v0,-32728(gp)
18cc:  24472040   addiu  a3,v0,8256
18d0:  8f828030   lw  v0,-32720(gp)
18d4:  24461e44   addiu  a2,v0,7748
18d8:  8fc50254   lw  a1,596(s8)
18dc:  8fc40250   lw  a0,592(s8)
18e0:  8f828090   lw  v0,-32624(gp)
18e4:  0040c825   move  t9,v0
18e8:  0320f809   jalr  t9
18ec:  00000000   nop
18f0:  8fdc0018   lw  gp,24(s8)
18f4:  afc20034   sw  v0,52(s8)
18f8:  8fc30034   lw  v1,52(s8)
18fc:  2402ffff   li  v0,-1
1900:  14620007   bne  v1,v0,1920 <main+0xe8>
1904:  00000000   nop
1908:  8fc30020   lw  v1,32(s8)
190c:  24020001   li  v0,1
1910:  10620057   beq  v1,v0,1a70 <main+0x238>
1914:  00000000   nop
1918:  10000052   b  1a64 <main+0x22c>
191c:  00000000   nop
1920:  8fc20034   lw  v0,52(s8)
1924:  2442ffaa   addiu  v0,v0,-86
1928:  2c43001a   sltiu  v1,v0,26
192c:  10600044   beqz  v1,1a40 <main+0x208>
1930:  00000000   nop
1934:  00021880   sll  v1,v0,0x2
1938:  8f828030   lw  v0,-32720(gp)
193c:  24421f80   addiu  v0,v0,8064
1940:  00621021   addu  v0,v1,v0
1944:  8c420000   lw  v0,0(v0)
```

```
1948:  005c1021   addu  v0,v0,gp
194c:  00400008   jr  v0
1950:  00000000   nop
1954:  24020003   li  v0,3
1958:  afc20020   sw  v0,32(s8)
195c:  1000003f   b  1a5c <main+0x224>
1960:  00000000   nop
1964:  24020002   li  v0,2
1968:  afc20020   sw  v0,32(s8)
196c:  1000003b   b  1a5c <main+0x224>
1970:  00000000   nop
1974:  8f828068   lw  v0,-32664(gp)
1978:  8c500000   lw  s0,0(v0)
197c:  8f828068   lw  v0,-32664(gp)
1980:  8c420000   lw  v0,0(v0)
1984:  00402025   move  a0,v0
1988:  8f82808c   lw  v0,-32628(gp)
198c:  0040c825   move  t9,v0
1990:  0320f809   jalr  t9
1994:  00000000   nop
1998:  8fdc0018   lw  gp,24(s8)
199c:  00401825   move  v1,v0
19a0:  27c20038   addiu  v0,s8,56
19a4:  00603025   move  a2,v1
19a8:  02002825   move  a1,s0
19ac:  00402025   move  a0,v0
19b0:  8f8280a0   lw  v0,-32608(gp)
19b4:  0040c825   move  t9,v0
19b8:  0320f809   jalr  t9
19bc:  00000000   nop
19c0:  8fdc0018   lw  gp,24(s8)
19c4:  27c20038   addiu  v0,s8,56
19c8:  afc20024   sw  v0,36(s8)
19cc:  10000023   b  1a5c <main+0x224>
19d0:  00000000   nop
19d4:  8f828068   lw  v0,-32664(gp)
19d8:  8c500000   lw  s0,0(v0)
19dc:  8f828068   lw  v0,-32664(gp)
19e0:  8c420000   lw  v0,0(v0)
19e4:  00402025   move  a0,v0
19e8:  8f82808c   lw  v0,-32628(gp)
19ec:  0040c825   move  t9,v0
19f0:  0320f809   jalr  t9
19f4:  00000000   nop
19f8:  8fdc0018   lw  gp,24(s8)
19fc:  00401825   move  v1,v0
1a00:  27c20138   addiu  v0,s8,312
1a04:  00603025   move  a2,v1
1a08:  02002825   move  a1,s0
1a0c:  00402025   move  a0,v0
1a10:  8f8280a0   lw  v0,-32608(gp)
1a14:  0040c825   move  t9,v0
1a18:  0320f809   jalr  t9
1a1c:  00000000   nop
```

```
1160    1a20:  8fdc0018   lw  gp,24(s8)
1161    1a24:  27c20138   addiu  v0,s8,312
1162    1a28:  afc20028   sw  v0,40(s8)
1163    1a2c:  1000000b   b  1a5c <main+0x224>
1164    1a30:  00000000   nop
1165    1a34:  afc00020   sw  zero,32(s8)
1166    1a38:  10000008   b  1a5c <main+0x224>
1167    1a3c:  00000000   nop
1168    1a40:  8f828030   lw  v0,-32720(gp)
1169    1a44:  24441e4c   addiu  a0,v0,7756
1170    1a48:  8f82803c   lw  v0,-32708(gp)
1171    1a4c:  0040c825   move  t9,v0
1172    1a50:  0411ff43   bal  1760 <imprimir_salida>
1173    1a54:  00000000   nop
1174    1a58:  8fdc0018   lw  gp,24(s8)
1175    1a5c:  1000ff97   b  18bc <main+0x84>
1176    1a60:  00000000   nop
1177    1a64:  8fc20020   lw  v0,32(s8)
1178    1a68:  1440004f   bnez  v0,1ba8 <main+0x370>
1179    1a6c:  00000000   nop
1180    1a70:  8f828078   lw  v0,-32648(gp)
1181    1a74:  8c420000   lw  v0,0(v0)
1182    1a78:  afc2002c   sw  v0,44(s8)
1183    1a7c:  8f828080   lw  v0,-32640(gp)
1184    1a80:  8c420000   lw  v0,0(v0)
1185    1a84:  afc20030   sw  v0,48(s8)
1186    1a88:  8fc20024   lw  v0,36(s8)
1187    1a8c:  10400018   beqz  v0,1af0 <main+0x2b8>
1188    1a90:  00000000   nop
1189    1a94:  27c30038   addiu  v1,s8,56
1190    1a98:  8f828030   lw  v0,-32720(gp)
1191    1a9c:  24451e88   addiu  a1,v0,7816
1192    1aa0:  00602025   move  a0,v1
1193    1aa4:  8f8280ac   lw  v0,-32596(gp)
1194    1aa8:  0040c825   move  t9,v0
1195    1aac:  0320f809   jalr  t9
1196    1ab0:  00000000   nop
1197    1ab4:  8fdc0018   lw  gp,24(s8)
1198    1ab8:  afc20030   sw  v0,48(s8)
1199    1abc:  8fc20030   lw  v0,48(s8)
1200    1ac0:  1440000b   bnez  v0,1af0 <main+0x2b8>
1201    1ac4:  00000000   nop
1202    1ac8:  8f828030   lw  v0,-32720(gp)
1203    1acc:  24441e8c   addiu  a0,v0,7820
1204    1ad0:  8f828040   lw  v0,-32704(gp)
1205    1ad4:  0040c825   move  t9,v0
1206    1ad8:  0411ff3c   bal  17cc <imprimir_error>
1207    1adc:  00000000   nop
1208    1ae0:  8fdc0018   lw  gp,24(s8)
1209    1ae4:  24020001   li  v0,1
1210    1ae8:  10000048   b  1c0c <main+0x3d4>
1211    1aec:  00000000   nop
1212    1af0:  8fc20028   lw  v0,40(s8)
1213    1af4:  1040000b   beqz  v0,1b24 <main+0x2ec>
```

```
1214    1af8:  00000000   nop
1215    1afc:  27c30138   addiu  v1,s8,312
1216    1b00:  8f828030   lw  v0,-32720(gp)
1217    1b04:  24451eac   addiu  a1,v0,7852
1218    1b08:  00602025   move  a0,v1
1219    1b0c:  8f8280ac   lw  v0,-32596(gp)
1220    1b10:  0040c825   move  t9,v0
1221    1b14:  0320f809   jalr  t9
1222    1b18:  00000000   nop
1223    1b1c:  8fdc0018   lw  gp,24(s8)
1224    1b20:  afc2002c   sw  v0,44(s8)
1225    1b24:  8fc20020   lw  v0,32(s8)
1226    1b28:  1040000a   beqz  v0,1b54 <main+0x31c>
1227    1b2c:  00000000   nop
1228    1b30:  8fc5002c   lw  a1,44(s8)
1229    1b34:  8fc40030   lw  a0,48(s8)
1230    1b38:  8f828044   lw  v0,-32700(gp)
1231    1b3c:  0040c825   move  t9,v0
1232    1b40:  0411fea7   bal  15e0 <encodeFileToBase64>
1233    1b44:  00000000   nop
1234    1b48:  8fdc0018   lw  gp,24(s8)
1235    1b4c:  10000008   b  1b70 <main+0x338>
1236    1b50:  00000000   nop
1237    1b54:  8fc5002c   lw  a1,44(s8)
1238    1b58:  8fc40030   lw  a0,48(s8)
1239    1b5c:  8f828048   lw  v0,-32696(gp)
1240    1b60:  0040c825   move  t9,v0
1241    1b64:  0411fecd   bal  169c <decodeFileFromBase64>
1242    1b68:  00000000   nop
1243    1b6c:  8fdc0018   lw  gp,24(s8)
1244    1b70:  8fc40030   lw  a0,48(s8)
1245    1b74:  8f828070   lw  v0,-32656(gp)
1246    1b78:  0040c825   move  t9,v0
1247    1b7c:  0320f809   jalr  t9
1248    1b80:  00000000   nop
1249    1b84:  8fdc0018   lw  gp,24(s8)
1250    1b88:  8fc4002c   lw  a0,44(s8)
1251    1b8c:  8f828070   lw  v0,-32656(gp)
1252    1b90:  0040c825   move  t9,v0
1253    1b94:  0320f809   jalr  t9
1254    1b98:  00000000   nop
1255    1b9c:  8fdc0018   lw  gp,24(s8)
1256    1ba0:  10000019   b  1c08 <main+0x3d0>
1257    1ba4:  00000000   nop
1258    1ba8:  8fc30020   lw  v1,32(s8)
1259    1bac:  24020002   li  v0,2
1260    1bb0:  1462000a   bne  v1,v0,1bdc <main+0x3a4>
1261    1bb4:  00000000   nop
1262    1bb8:  8f828030   lw  v0,-32720(gp)
1263    1bbc:  24441eb0   addiu  a0,v0,7856
1264    1bc0:  8f82803c   lw  v0,-32708(gp)
1265    1bc4:  0040c825   move  t9,v0
1266    1bc8:  0411fee5   bal  1760 <imprimir_salida>
1267    1bcc:  00000000   nop
```

```
    1bd0:  8fdc0018   lw  gp,24(s8)
    1bd4:  1000000c   b  1c08 <main+0x3d0>
    1bd8:  00000000   nop
    1bdc:  8fc30020   lw  v1,32(s8)
    1be0:  24020003   li  v0,3
    1be4:  14620008   bne  v1,v0,1c08 <main+0x3d0>
    1be8:  00000000   nop
    1bec:  8f828030   lw  v0,-32720(gp)
    1bf0:  24441f70   addiu  a0,v0,8048
    1bf4:  8f82803c   lw  v0,-32708(gp)
    1bf8:  0040c825   move  t9,v0
    1bfc:  0411fed8   bal  1760 <imprimir_salida>
    1c00:  00000000   nop
    1c04:  8fdc0018   lw  gp,24(s8)
    1c08:  00001025   move  v0,zero
    1c0c:  03c0e825   move  sp,s8
    1c10:  8fbf024c   lw  ra,588(sp)
    1c14:  8fbe0248   lw  s8,584(sp)
    1c18:  8fb00244   lw  s0,580(sp)
    1c1c:  27bd0250   addiu  sp,sp,592
    1c20:  03e00008   jr  ra
    1c24:  00000000   nop
  ...

00001c30 <__libc_csu_init>:
    1c30:  3c1c0002   lui  gp,0x2
    1c34:  279c8460   addiu  gp,gp,-31648
    1c38:  0399e021   addu  gp,gp,t9
    1c3c:  27bdffc8   addiu  sp,sp,-56
    1c40:  8f99804c   lw  t9,-32692(gp)
    1c44:  afbc0010   sw  gp,16(sp)
    1c48:  afb50030   sw  s5,48(sp)
    1c4c:  00c0a825   move  s5,a2
    1c50:  afb4002c   sw  s4,44(sp)
    1c54:  00a0a025   move  s4,a1
    1c58:  afb30028   sw  s3,40(sp)
    1c5c:  00809825   move  s3,a0
    1c60:  afb20024   sw  s2,36(sp)
    1c64:  afb0001c   sw  s0,28(sp)
    1c68:  afbf0034   sw  ra,52(sp)
    1c6c:  0411fb0c   bal  8a0 <_init>
    1c70:  afb10020   sw  s1,32(sp)
    1c74:  8fbc0010   lw  gp,16(sp)
    1c78:  8f908050   lw  s0,-32688(gp)
    1c7c:  8f928054   lw  s2,-32684(gp)
    1c80:  02509023   subu  s2,s2,s0
    1c84:  00129083   sra  s2,s2,0x2
    1c88:  12400009   beqz  s2,1cb0 <__libc_csu_init+0x80>
    1c8c:  00008825   move  s1,zero
    1c90:  8e190000   lw  t9,0(s0)
    1c94:  26310001   addiu  s1,s1,1
    1c98:  02a03025   move  a2,s5
    1c9c:  02802825   move  a1,s4
    1ca0:  0320f809   jalr  t9
```

```
1322      1ca4:  02602025   move  a0,s3
1323      1ca8:  1651fff9   bne  s2,s1,1c90 <__libc_csu_init+0x60>
1324      1cac:  26100004   addiu  s0,s0,4
1325      1cb0:  8fbf0034   lw  ra,52(sp)
1326      1cb4:  8fb50030   lw  s5,48(sp)
1327      1cb8:  8fb4002c   lw  s4,44(sp)
1328      1cbc:  8fb30028   lw  s3,40(sp)
1329      1cc0:  8fb20024   lw  s2,36(sp)
1330      1cc4:  8fb10020   lw  s1,32(sp)
1331      1cc8:  8fb0001c   lw  s0,28(sp)
1332      1ccc:  03e00008   jr  ra
1333      1cd0:  27bd0038   addiu  sp,sp,56
1334
1335 00001cd4 <__libc_csu_fini>:
1336      1cd4:  03e00008   jr  ra
1337      1cd8:  00000000   nop
1338      1cdc:  00000000   nop
1339
1340 Disassembly of section .MIPS.stubs:
1341
1342 00001ce0 <_MIPS_STUBS_>:
1343      1ce0:  8f998010   lw  t9,-32752(gp)
1344      1ce4:  03e07825   move  t7,ra
1345      1ce8:  0320f809   jalr  t9
1346      1cec:  24180021   li  t8,33
1347      1cf0:  8f998010   lw  t9,-32752(gp)
1348      1cf4:  03e07825   move  t7,ra
1349      1cf8:  0320f809   jalr  t9
1350      1cfc:  2418001f   li  t8,31
1351      1d00:  8f998010   lw  t9,-32752(gp)
1352      1d04:  03e07825   move  t7,ra
1353      1d08:  0320f809   jalr  t9
1354      1d0c:  2418001e   li  t8,30
1355      1d10:  8f998010   lw  t9,-32752(gp)
1356      1d14:  03e07825   move  t7,ra
1357      1d18:  0320f809   jalr  t9
1358      1d1c:  2418001d   li  t8,29
1359      1d20:  8f998010   lw  t9,-32752(gp)
1360      1d24:  03e07825   move  t7,ra
1361      1d28:  0320f809   jalr  t9
1362      1d2c:  2418001c   li  t8,28
1363      1d30:  8f998010   lw  t9,-32752(gp)
1364      1d34:  03e07825   move  t7,ra
1365      1d38:  0320f809   jalr  t9
1366      1d3c:  2418001a   li  t8,26
1367      1d40:  8f998010   lw  t9,-32752(gp)
1368      1d44:  03e07825   move  t7,ra
1369      1d48:  0320f809   jalr  t9
1370      1d4c:  24180019   li  t8,25
1371      1d50:  8f998010   lw  t9,-32752(gp)
1372      1d54:  03e07825   move  t7,ra
1373      1d58:  0320f809   jalr  t9
1374      1d5c:  24180018   li  t8,24
1375      1d60:  8f998010   lw  t9,-32752(gp)
```

```
    1d64:  03e07825   move  t7,ra
    1d68:  0320f809   jalr  t9
    1d6c:  24180017   li  t8,23
    1d70:  8f998010   lw  t9,-32752(gp)
    1d74:  03e07825   move  t7,ra
    1d78:  0320f809   jalr  t9
    1d7c:  24180015   li  t8,21
    1d80:  8f998010   lw  t9,-32752(gp)
    1d84:  03e07825   move  t7,ra
    1d88:  0320f809   jalr  t9
    1d8c:  24180012   li  t8,18
    1d90:  8f998010   lw  t9,-32752(gp)
    1d94:  03e07825   move  t7,ra
    1d98:  0320f809   jalr  t9
    1d9c:  2418000f   li  t8,15
  ...

Disassembly of section .fini:

00001db0 <_fini>:
    1db0:  3c1c0002   lui  gp,0x2
    1db4:  279c82e0   addiu  gp,gp,-32032
    1db8:  0399e021   addu  gp,gp,t9
    1dbc:  27bdffe0   addiu  sp,sp,-32
    1dc0:  afbc0010   sw  gp,16(sp)
    1dc4:  afbf001c   sw  ra,28(sp)
    1dc8:  8fbf001c   lw  ra,28(sp)
    1dcc:  03e00008   jr  ra
    1dd0:  27bd0020   addiu  sp,sp,32
```

7.3. Enunciado

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`