



**FACULTAD
DE INGENIERIA**

Universidad de Buenos Aires

[66.20] ORGANIZACIÓN DE COMPUTADORAS

TRABAJO PRÁCTICO 0

2^{DO} CUATRIMESTRE 2018

Infraestructura básica

AUTORES

Husain, Ignacio Santiago. - #90.117

<santiago.husain@gmail.com>

Pesado, Lucía. - #98.275

<luupesado@gmail.com>

Verstraeten, Federico. - #92.892

<federico.verstraeten@gmail.com>

CÁTEDRA

Dr. Ing. Hamkalo, José Luis.

CURSO

Ing. Santi, Leandro.

FECHA DE ENTREGA

xx/xx/2018

FECHA DE APROBACIÓN

CALIFICACIÓN

FIRMA DE APROBACIÓN

Índice

1. Enunciado del trabajo práctico	2
2. Objetivos	6
3. Diseño e implementación del programa	6
4. Compilación del programa	8
5. Pruebas	9
5.1. Pruebas en las opciones de programa	9
5.1.1. Pruebas de codificación y decodificación	12
5.1.2. Tiempos de codificación y decodificación	13
6. Herramientas de hardware y software utilizadas	15
7. Conclusiones	15
Referencias	16
A. Makefile	17
A.0.1. makefile	17
B. Tests	19
B.0.1. runTests.sh	19
C. Código fuente	32
C.0.1. main.c	32
C.0.2. common.h	34
C.0.3. decoder.h	36
C.0.4. decoder.c	38
C.0.5. encoder.h	42
C.0.6. encoder.c	44
C.0.7. parser.h	48
C.0.8. parser.c	50
C.0.9. messages.h	55
C.0.10. main.s	57
C.0.11. parser.s	60
C.0.12. encoder.s	76
C.0.13. decoder.s	84

1. Enunciado del trabajo práctico

66:20 Organización de Computadoras Trabajo práctico #0: Infraestructura básica 2^{do} cuatrimestre de 2018

\$Date: 2018/09/08 23:16:30 \$

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 6), 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 grupo¹, y se valorarán aquellos escritos usando la herramienta \TeX / \LaTeX .

4. Recursos

Usaremos el programa GXemul [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 NetBSD [2].

En la clase del 21/8 hemos repasado los pasos necesarios para la instalación y configuración del entorno de desarrollo.

¹<http://groups.yahoo.com/group/orga-comp>

5. Programa

Se trata de escribir, en lenguaje C, un programa para codificar y decodificar información en formato base 64: el programa recibirá, por línea de comando, los archivos o *streams* de entrada y salida, y la acción a realizar, codificar (acción por defecto) o decodificar. De no recibir los nombres de los archivos (o en caso de recibir – como nombre de archivo) usaremos los *streams* estándar, **stdin** y **stdout**, según corresponda. A continuación, iremos leyendo los datos de la entrada, generando la salida correspondiente. De ocurrir errores, usaremos **stderr**. Una vez agotados los datos de entrada, el programa debe finalizar adecuadamente, retornando al sistema operativo.

Estrictamente hablando, base 64 es un grupo de esquemas de codificación similares. En nuestra implementación, estaremos siguiendo particularmente el esquema establecido en [3], con el siguiente agregado: si se recibe una secuencia de caracteres inválida en la decodificación, debe asumirse como una condición de error que el programa deberá reportar adecuadamente y detener el procesamiento en ese punto.

5.1. 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.
    -i, --input         Location of the input file.
    -o, --output        Location of the output file.
    -a, --action        Program action: encode (default) or decode.
Examples:
    tp0 -a encode -i ~/input -o ~/output
    tp0 -a decode
```

Codificamos un archivo vacío (cantidad de bytes nula):

```
$ touch /tmp/zero.txt
$ tp0 -a encode -i /tmp/zero.txt -o /tmp/zero.txt.b64
$ ls -l /tmp/zero.txt.b64
-rw-r--r--  1 user group 0 2018-09-08 16:21 /tmp/zero.txt.b64
```

Codificamos el carácter ASCII M,

```
$ echo -n M | tp0
TQ==
```

Codificamos los caracteres ASCII M y a,

```
$ echo -n Ma | tp0
TWE=
```

Codificamos M a n,

```
$ echo -n Man | tp0
TWFu
```

Codificamos y decodificamos:

```
$ echo Man | tp0 | tp0 -a decode
Man
```

Verificamos bit a bit:

```
$ echo xyz | tp0 | tp0 -a decode | od -t c
0000000  x  y  z  \n
0000004
```

Codificamos 1024 bytes, para verificar que el programa genere líneas de no mas de 76 unidades de longitud:

```
$ yes | head -c 1024 | tp0 -a encode
eQp5CnkKeQp5CnkKeQp5CnkKeQp5CnkKeQp5CnkKeQp5CnkKeQp5CnkKeQp5CnkK
...
eQp5CnkKeQp5CnkKeQp5CnkKeQp5CnkKeQp5CnkKeQp5CnkKeQp5CnkKeQp5Cg==
```

Verificamos que la cantidad de bytes decodificados, sea 1024:

```
$ yes | head -c 1024 | tp0 -a encode | tp0 -a decode | wc -c
1024
```

Generamos archivos de tamaño creciente, y verificamos que el procesamiento de nuestro programa no altere los datos:

```
$ n=1;
$ while ;; do
>   head -c $n </dev/urandom >/tmp/in.bin;
>   tp0 -a encode -i /tmp/in.bin -o /tmp/out.b64;
>   tp0 -a decode -i /tmp/out.b64 -o /tmp/out.bin;
>   if diff /tmp/in.bin /tmp/out.bin; then ;; else
>       echo ERROR: $n;
>       break;
>   fi
>   echo ok: $n;
>   n="`expr $n + 1`";
>   rm -f /tmp/in.bin /tmp/out.b64 /tmp/out.bin
> done
ok: 1
ok: 2
ok: 3
...
```

6. Informe

El informe deberá incluir al menos las siguientes secciones:

- Documentación relevante al diseño e implementación del programa;
- Comando(s) para compilar el programa;
- Las corridas de prueba, con los comentarios pertinentes;
- El código fuente, en lenguaje C, el cual también deberá entregarse en formato digital compilable (incluyendo archivos de entrada y salida de pruebas);
- El código MIPS32 generado por el compilador;
- Este enunciado.

El informe deberá entregarse en formato impreso y digital.

Referencias

- [1] GXemul, <http://gavare.se/gxemul/>.
- [2] The NetBSD project, <http://www.netbsd.org/>.
- [3] RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies; sección 6.8, Base64 Content-Transfer-Encoding. <http://tools.ietf.org/html/rfc2045#section-6.8>.
- [4] Base64 (Wikipedia). <http://en.wikipedia.org/wiki/Base64>.

2. Objetivos

El presente trabajo tiene los siguientes objetivos:

- Diseñar un codificador/decodificador (*codec*) de información en formato base 64 utilizando el lenguaje de programación C.
- Compilar dicho codec en el sistema operativo netBSD en una máquina con arquitectura MIPS32, y producir el código assembly MIPS32 del mismo.
- Realizar pruebas de caja negra para verificar que el programa está funcionando de manera correcta.
- Utilizar la infraestructura básica que será utilizada en trabajos posteriores para programar en assembly MIPS32.

3. Diseño e implementación del programa

Se diseñó un programa en lenguaje ANSI C que implementa el codec descrito en el documento RFC2045 *Base64 Content-Transfer-Encoding* [4], [5], con la diferencia de si en la etapa de decodificación se encuentra un caracter no válido, la ejecución del programa se suspende, notificando al usuario del error producido.

El programa se estructura de la siguiente manera:

- Análisis gramatical de la línea de comandos: se analizan las opciones ingresadas por la línea de comandos haciendo uso de la función `parseCmdline()`. La misma se encarga de inicializar una estructura del tipo `params_t` utilizada para almacenar las opciones que ingresó el cliente, y cuya definición es

```
1      typedef struct params_t
2      {
3          char *action;
4          FILE *inputStream;
5          FILE *outputStream;
6      } params_t;
```

Además, hace uso de la función `getopt_long()` de la biblioteca `getopt.h`. Dicha función provee una forma simple de procesar cada opción que es leída, extrayendo los argumentos de cada una. En caso de que no se encuentre alguna opción, se utiliza su valor por defecto según las especificaciones del trabajo.

- Validación de opciones: a medida que se va analizando cada opción de la línea de comandos, se valida cada una de ellas utilizando las funciones

```
1      outputCode optAction()
2      outputCode optOutput()
```

```
3     outputCode optInput()  
4     void optHelp()  
5     void optVersion()
```

Además, las mismas realizan la correcta inicialización de las diferentes variables dentro de la estructura `params_t` descrita en el punto anterior, o en caso de que el usuario ingresó las opciones de ayuda e indicación de versión del programa, se imprime por el flujo `stderr` dicha información.

En caso de que se encuentre algún error en el argumento de alguna de las opciones, se lo informa al usuario por el flujo `stderr` de dicho error, y se aborta la ejecución del programa utilizando la función `exit()`. Para ello, se creó un tipo enumerativo para simplificar el manejo de errores, definiendo los códigos que pueden devolver las funciones desarrolladas:

```
1     typedef enum outputCodes_ {  
2         outOK ,  
3         outERROR  
4     } outputCode;
```

- **Codificación/Decodificación:** una vez realizada la validación de las opciones y carga de configuración del programa, se utilizan las funciones `base256ToBase64()` o `base64ToBase256()` para codificar o decodificar el flujo de entrada, dependiendo de qué fue lo que el cliente le solicitó al programa.

El proceso de codificación se realiza tomando cada caracter de entrada, y guardando los dos bits más y menos significativos para poder concatenarlos con los próximos caracteres que vayan ingresando por el flujo de entrada. Se realiza la extracción de los bits utilizando una máscara de bits y desplazándola adecuadamente dependiendo cuántos bytes se fueron leyendo. Con esta información, se extrae el índice de la tabla de codificación, y se almacenan los caracteres codificados en un arreglo de 4 bytes que posteriormente son impresos en el flujo de salida especificado. Cuando por el flujo de entrada ingresa “EOF”, entonces se incluye el caracter de padding “=”, dependiendo de cuántos caracteres hayan sido codificados en los pasos previos para que el arreglo de salida quede con 4 bytes completos.

Por otro lado, la función `encode()` se encarga no solo de codificar el flujo de entrada, si no de llevar un registro de la cantidad de caracteres codificados para incluir un salto de línea cada 76 caracteres como se establece en la especificación del codec.

El proceso de decodificación se realiza en tres pasos y de a bloques de 4 bytes. Para cada caracter a decodificar, primero se busca en la tabla traducción `translationTableB64` el índice correspondiente al caracter codificado que ingresó por el flujo de entrada. Luego se lo posiciona dentro de un entero de 4 bytes, en una posición que depende de cuál de los 4 bytes del bloque de entrada fue leído. Por último, se utiliza una máscara de bits para ir extrayendo cada uno de los caracteres de 1 byte correspondientes, y

guarándolos en un array de salida, que será escrito en el flujo de salida posteriormente por la función `decode()`.

- Terminación del programa: una vez finalizada la codificación o decodificación, se cierran los flujos de entrada y salida, y se retorna al sistema operativo.

El código fuente se encuentra en el apéndice, tanto en lenguaje C (C.0.1) como en MIPS32 (C.0.10).

4. Compilación del programa

Debido al requerimiento de utilizar el programa en una computadora con arquitectura MIPS32, se utiliza el emulador `GXemu1` que provee la cátedra, utilizando una máquina virtual que contiene el sistema operativo NetBSD con las herramientas necesarias para compilar el programa desarrollado: `gcc` y `make`.

Para obtener un ejecutable, se creó un archivo `makefile` cuyo contenido se puede ver en la sección A.0.1. Para ejecutarlo, posicionarse en el directorio `src/` y ejecutar el siguiente comando:

```
1 $ make
```

donde el mismo generará el ejecutable con nombre `tp0`. En caso de requerir que el código assembly MIPS32 de cada archivo con código fuente, ejecutar el siguiente comando:

```
1 $ make assembly ARGS=-mrnames
```

5. Pruebas

Según Patton en [2], una prueba de caja negra (o *black-box test*) es aquella donde se realizan pruebas al programa sin acceder al código fuente. Es decir, la única información a la que se tiene acceso es aquella que define el comportamiento del programa según las especificaciones del cliente. En esta sección se muestran los resultados de las distintas pruebas de caja negra que se realizaron sobre el programa para determinar su robustez y fiabilidad ante diferentes tipos de entradas.

Se creó un script en Bash para automatizar las pruebas del programa. El código del script se encuentra en la sección B.0.1, y está compuesto por 16 tests más 2 para medir tiempos de ejecución. Dado que el programa en este caso solo retorna un solo código, el script devuelve un mensaje de éxito, indicando la correcta ejecución del programa y que se generó correctamente el archivo de salida.

La salida del script se divide en 2 secciones, cada una con un encabezado indicando el inicio del nuevo test y su nombre, y varias líneas por cada test. La primer línea del test es el comando ejecutado, indicado con la etiqueta `Testing:`. La segunda indica si el test fue exitoso o no mediante la etiqueta `PASSED/FAILED`, y las siguientes líneas son los resultados que produce el programa (mensajes de error, etc...). Por ejemplo, para la prueba de la opción “-i”, se tiene lo siguiente:

```

1 -----
2 TEST1: inexistent 'input' stream.
3 -----
4 Testing: ./tp0 -i 1
5 PASSED
6   PROGRAM OUTPUT:
7     ERROR: Can't open input stream.
```

donde se ve que el test fue satisfactorio ya que se introdujo un nombre de flujo de entrada inválido.

Para ejecutar el script con las pruebas, posicionarse en el directorio `src` y ejecutar el comando

`.runTests.sh` en la terminal de Linux.

5.1. Pruebas en las opciones de programa

En el script de tests se prueban diferentes combinaciones de las opciones de entrada para verificar si el programa es capaz de detectar errores. Los 7 primeros son validaciones utilizando opciones y parámetros inválidos, donde se verifica que al intentar ejecutarlo, el programa termine y retorne un mensaje que indique el motivo de la ejecución fallida. El test `test4_valid_parameters` se corresponde con ejecuciones que retornan un código de éxito.

Las salidas arrojadas por el script fueron las siguientes:

```

1 -----
2 TEST1: inexistent 'input' stream.
```

```
3 -----
4 Testing: ./tp0 -i 1
5 PASSED
6   PROGRAM OUTPUT:
7     ERROR: Can't open input stream.
8 Testing: ./tp0 -i hola.bin
9 PASSED
10  PROGRAM OUTPUT:
11    ERROR: Can't open input stream.
12 -----
13 TEST11: no 'input' option parameters.
14 -----
15 Testing: ./tp0 -i
16 PASSED
17  PROGRAM OUTPUT:
18    ./tp0: option requires an argument -- 'i'
19 -----
20 TEST12: invalid 'input' stream.
21 -----
22 Testing: ./tp0 -i .
23 PASSED
24  PROGRAM OUTPUT:
25    ERROR: Invalid input stream.
26 Testing: ./tp0 -i ..
27 PASSED
28  PROGRAM OUTPUT:
29    ERROR: Invalid input stream.
30 Testing: ./tp0 -i /
31 PASSED
32  PROGRAM OUTPUT:
33    ERROR: Invalid input stream.
34 Testing: ./tp0 -i //
35 PASSED
36  PROGRAM OUTPUT:
37    ERROR: Invalid input stream.
38 -----
39 TEST2: invalid 'output' stream.
40 -----
41 Testing: ./tp0 -i ../tests/leviathan.input -o .
42 PASSED
43  PROGRAM OUTPUT:
44    ERROR: Invalid output stream.
45 Testing: ./tp0 -i ../tests/leviathan.input -o ..
46 PASSED
47  PROGRAM OUTPUT:
48    ERROR: Invalid output stream.
49 Testing: ./tp0 -i ../tests/leviathan.input -o /
```

```
50 PASSED
51   PROGRAM OUTPUT:
52     ERROR: Invalid output stream.
53 Testing: ./tp0 -i ../tests/leviathan.input -o //
54 PASSED
55   PROGRAM OUTPUT:
56     ERROR: Invalid output stream.
57 -----
58 TEST21: no 'output' option parameters.
59 -----
60 Testing: ./tp0 -o
61 PASSED
62   PROGRAM OUTPUT:
63     ./tp0: option requires an argument -- 'o'
64 -----
65 TEST3: invalid 'action' parameters.
66 -----
67 Testing: ./tp0 -a bad_action
68 PASSED
69   PROGRAM OUTPUT:
70     ERROR: Invalid action argument.
71 Testing: ./tp0 -a 1
72 PASSED
73   PROGRAM OUTPUT:
74     ERROR: Invalid action argument.
75 Testing: ./tp0 -a .
76 PASSED
77   PROGRAM OUTPUT:
78     ERROR: Invalid action argument.
79 Testing: ./tp0 -a ..
80 PASSED
81   PROGRAM OUTPUT:
82     ERROR: Invalid action argument.
83 Testing: ./tp0 -a /
84 PASSED
85   PROGRAM OUTPUT:
86     ERROR: Invalid action argument.
87 Testing: ./tp0 -a //
88 PASSED
89   PROGRAM OUTPUT:
90     ERROR: Invalid action argument.
91 Testing: ./tp0 -a $
92 PASSED
93   PROGRAM OUTPUT:
94     ERROR: Invalid action argument.
95 -----
96 TEST31: no 'action' option parameters.
```

```

97 -----
98 Testing: ./tp0 -a
99 PASSED
100     PROGRAM OUTPUT:
101     ./tp0: option requires an argument -- 'a'
102 -----
103 TEST4: all options with correct parameters.
104 -----
105 Testing: ./tp0 -a encode -i ../tests/test1.bin -o ../tests/↵
106     test1_out.bin
107 PASSED
108     PROGRAM OUTPUT:
109 Testing: ./tp0 -a decode -i ../tests/test1_out.bin -o ../tests/↵
110     /test2.bin
111 PASSED
112     PROGRAM OUTPUT:

```

5.1.1. Pruebas de codificación y decodificación

Los tests cuyo nombre son del estilo test5x_IO_validation se crearon para probar si el codec funciona correctamente, ya sea ingresando palabras y/o archivos cuya codificación es conocida, como también archivos aleatorios que son codificados y decodificados para probar si el programa realiza una operación de identidad.

Además, la prueba test57_IO_validation verifica que el programa produzca saltos de línea cada 76 caracteres según la especificación del enunciado, y además que si se codificaron N caracteres, la decodificación produzca N caracteres.

Las salidas del script para dichas pruebas fue la siguiente:

```

1 -----
2 TEST5: input-output should be the same.
3 -----
4 PASSED:  n = 1
5 PASSED:  n = 2
6 PASSED:  n = 3
7 PASSED:  n = 4
8 PASSED:  n = 5
9 PASSED:  n = 6
10 PASSED:  n = 7
11 PASSED:  n = 8
12 PASSED:  n = 9
13 PASSED:  n = 10
14 -----
15 TEST51: input known text with known encoding.
16 -----
17 PASSED:  No differences.

```

```

18 -----
19 TEST52: Encode letter 'M'.
20 -----
21 PASSED:  No differences.
22 -----
23 TEST53: Encode 'Ma'.
24 -----
25 PASSED:  No differences.
26 -----
27 TEST54: Encode 'Man'.
28 -----
29 PASSED:  No differences.
30 -----
31 TEST55: Encode and decode 'Man'.
32 -----
33 PASSED:  No differences.
34 -----
35 TEST56: Check bit by bit.
36 -----
37 PASSED:  No differences.
38 -----
39 TEST57: Check max line length and number of encoded bytes.
40 -----
41 PASSED:  No differences in line count.
42 PASSED:  No differences in word count.

```

5.1.2. Tiempos de codificación y decodificación

Los tests `test6_encoding_execution_times` y `test7_decoding_execution_times` se utilizan para evaluar los tiempos de ejecución del programa completo cuando se codifican y decodifican archivos binarios aleatorios de varios tamaños.

Las salidas de los mismos se muestran a continuación, donde la letra `n` indica la cantidad de bytes del archivo binario.

```

1 -----
2 TEST6: encoding execution times.
3 -----
4 n: 1          1,00 [ms]
5 n: 2          2,00 [ms]
6 n: 4          2,00 [ms]
7 n: 8          1,00 [ms]
8 n: 16         1,00 [ms]
9 n: 32         1,00 [ms]
10 n: 64         1,00 [ms]
11 n: 128        2,00 [ms]
12 n: 256        1,00 [ms]

```

```
13 n: 512 6,00 [ms]
14 n: 1024 2,00 [ms]
15 n: 2048 1,00 [ms]
16 n: 4096 4,00 [ms]
17 n: 8192 5,00 [ms]
18 n: 16384 2,00 [ms]
19 n: 32768 2,00 [ms]
20 n: 65536 6,00 [ms]
21 n: 131072 3,00 [ms]
22 n: 262144 3,00 [ms]
23 n: 524288 2,00 [ms]
24 n: 1048576 1,00 [ms]
25 n: 2097152 1,00 [ms]
26 n: 4194304 2,00 [ms]
27 n: 8388608 1,00 [ms]
28 -----
29 TEST7: decoding execution times.
30 -----
31 n: 1 1,00 [ms]
32 n: 2 1,00 [ms]
33 n: 4 1,00 [ms]
34 n: 8 1,00 [ms]
35 n: 16 1,00 [ms]
36 n: 32 1,00 [ms]
37 n: 64 2,00 [ms]
38 n: 128 2,00 [ms]
39 n: 256 2,00 [ms]
40 n: 512 2,00 [ms]
41 n: 1024 1,00 [ms]
42 n: 2048 2,00 [ms]
43 n: 4096 2,00 [ms]
44 n: 8192 1,00 [ms]
45 n: 16384 1,00 [ms]
46 n: 32768 1,00 [ms]
47 n: 65536 1,00 [ms]
48 n: 131072 2,00 [ms]
49 n: 262144 1,00 [ms]
50 n: 524288 1,00 [ms]
51 n: 1048576 1,00 [ms]
52 n: 2097152 1,00 [ms]
53 n: 4194304 1,00 [ms]
54 n: 8388608 1,00 [ms]
55 -----
56 Test suite ended.
57 -----
58 All tests passed.
```

6. Herramientas de hardware y software utilizadas

La computadora utilizada para realizar el desarrollo y las pruebas tiene las siguientes especificaciones:

- Procesador: Intel i3-6100.
- Memoria: 8GB RAM DDR4.
- Almacenamiento: Disco magnético de 200GB de 7200RPM.

Los programas se desarrollaron en el sistema operativo Linux Ubuntu, cuyos datos de distribución son

```
Distributor ID: Ubuntu
Description: Ubuntu 16.04.2 LTS
Release: 16.04
Codename: xenial
```

Además, se utilizaron las siguientes herramientas:

- Compilador del proyecto: GNU gcc (Ubuntu 5.4.0-6ubuntu1 16.04.4) 5.4.0 20160609 [7].
- Control del proceso de compilación: GNU Make 4.1 [6].
- Compilador del presente informe: Latex pdfTeX 3.14159265-2.6-1.40.16 (TeX Live 2015/Debian) [8].
- Edición de código fuente: VIM - Vi IMproved 7.4 (2013 Aug 10, compiled Nov 24 2016 16:44:48) y Atom 1.15.0 [9] [10].
- Depuración del programa: gdb [11].

7. Conclusiones

Se creó un programa en lenguaje C que permite codificar y decodificar información en formato base 64, cumpliendo con las especificaciones del enunciado del trabajo. Se describió el flujo del programa y las funciones y estructuras más importantes del mismo, junto con el manejo de errores. Además, se describió el modo de compilación del programa utilizando las herramientas gcc y make.

Por otro lado, se crearon casos de prueba de caja negra para verificar el correcto funcionamiento del programa. Las mismas cubren tests de validación de opciones del programa ingresando combinaciones de opciones válidas e inválidas para determinar si este es capaz de detectar errores. Además, se crearon pruebas de codificación y decodificación de archivos, tamaños máximos de líneas, y tiempos de ejecución.

De esta forma se deja lista la infraestructura básica para los siguientes proyectos donde se trabajará con la arquitectura MIPS32 y el consecuente análisis y desarrollo de código assembly MIPS32.

Referencias

- [1] Kernighan, B. W. - Ritchie, D. M. - *C Programming Language* - 2nd edition - Prentice Hall - 1988.
- [2] Patton, R. - *Software Testing* - 2nd edition - Sams Indianapolis, IN, USA 2005.
- [3] *Apuntes del curso 66.20 Organización de Computadoras* - Cátedra Hamkalo - Facultad de Ingeniería de la Universidad de Buenos Aires.
- [4] *RFC 2045* - <https://tools.ietf.org/html/rfc2045#section-6.8>
- [5] *Base 64* - <https://en.wikipedia.org/wiki/Base64>
- [6] *GNU Make* - <https://www.gnu.org/software/make/>
- [7] *GNU Gcc* - <https://gcc.gnu.org/>
- [8] *L^AT_EX* - <https://www.latex-project.org/>
- [9] *VIM* - <https://vim.sourceforge.io/>
- [10] *Atom* - <https://atom.io/>
- [11] *GNU gdb* - <https://www.gnu.org/software/gdb/>

A. Makefile

A.0.1. makefile

```
1 # -----
2 # @Title:   FIUBA - 66.20 Organizacion de Computadoras.
3 # @Project: TP0 - Infraestructura basica.
4 # -----
5 # @Filename: makefile
6 # -----
7 # @Authors:
8 #   Husain, Ignacio Santiago.
9 #       santiago.husain at gmail dot com
10 #   Pesado, Lucia.
11 #       luupesado at gmail dot com
12 #   Verstraeten, Federico.
13 #       federico.verstraeten at gmail dot com
14 #
15 # @Date:           07-Sep-2018 12:57:50 pm
16 # @Last modified by:  Ignacio Santiago Husain
17 # @Last modified time: 18-Sep-2018 6:43:10 pm
18 #
19 # @Copyright (C):
20 #   This file is part of 'TP0 - Infraestructura basica.'.
21 #   Unauthorized copying or use of this file via any medium
22 #   is strictly prohibited.
23 # -----
24 #
25 # The source files must have .c extension.
26 # The object code must have .o extension.
27 # The header files must have .h extension.
28 #
29 # To compile, execute 'make'.
30 # To clean all the compilation files, issue the command
31 # 'make clean'.
32 #
33 # -----
34 # List all the header and object files separated by a blank
35 # space.
36 _DEPS = messages.h common.h parser.h encoder.h decoder.h
37 _SRC = main.c parser.c encoder.c decoder.c
38 _OBJ = main.o parser.o encoder.o decoder.o
39 # -----
40 # Configuration.
41 CC = gcc
42 CFLAGS = -ansi -I. -O0
```

```
43 OUTPUT1 = tp0
44 # -----
45 all: $(OUTPUT1)
46
47 $(OUTPUT1): $(OBJ)
48     $(CC) $(CFLAGS) -o $(OUTPUT1) $(OBJ)
49
50 assembly:
51     $(CC) $(CFLAGS) -S $_SRC $(ARGS)
52
53 .PHONY: clean assembly
54
55 clean:
56     rm -f ./*.o *~ core ./*~ ./*.s
57     rm -f $(OUTPUT1)
```

B. Tests

B.0.1. runTests.sh

```
1 #!/bin/bash
2
3 # -----
4 # @Title:    FIUBA - 66.20 Organizacion de Computadoras.
5 # @Project: TP0 - Infraestructura basica.
6 # -----
7 # @Filename: runTests.sh
8 # -----
9 # @Authors:
10 #     Husain, Ignacio Santiago.
11 #         santiago.husain at gmail dot com
12 #     Pesado, Lucia.
13 #         luupesado at gmail dot com
14 #     Verstraeten, Federico.
15 #         federico.verstraeten at gmail dot com
16 #
17 # @Date:          07-Sep-2018 2:12:07 pm
18 # @Last modified by:  Ignacio Santiago Husain
19 # @Last modified time: 18-Sep-2018 5:18:48 pm
20 #
21 # @Copyright (C):
22 #     This file is part of 'TP0 - Infraestructura basica.'.
23 #     Unauthorized copying or use of this file via any medium
24 #     is strictly prohibited.
25 # -----
26 #
27 # Script to test errors in the program arguments.
28 #
29 # To remove newlines from a textfile, use
30 # printf %s "$(cat file)" > file
31 #
32 # To print contents of a file, including control characters, ↵
33 # do:
34 # oc -c file
35 #
36 # -----
37 # Program name to test.
38 PROGRAM_NAME='./tp0 '
39
40 # Failed tests counter.
41 failedTests=0;
```

```

42
43 # Colors to be used.
44 RED="\e[31m";
45 GREEN="\e[32m";
46 CYAN="\e[96m";
47 YELLOW="\e[93m";
48 BOLD="\033[1m";
49 DEFAULT="\e[0m";
50
51 # Helper and formatting functions definitions.
52 function header() {
53     echo -e "$CYAN↵
54         -----↵
55         $DEFAULT";
56     echo -e "$CYAN$1$DEFAULT";
57     echo -e "$CYAN↵
58         -----↵
59         $DEFAULT";
60 }
61
62 function msg_true() {
63     echo -e "$GREEN\OPASSED \n $DEFAULT PROGRAM OUTPUT:\n\t$1";
64 }
65
66 function msg_false() {
67     echo -e "$RED\OFAILED \n $DEFAULT PROGRAM OUTPUT:\n\t$1";
68 }
69
70 function msg_testing() {
71     echo -e "Testing: $BOLD$1$DEFAULT";
72 }
73
74 function success_msg() {
75     echo -e " $GREEN$1$DEFAULT";
76 }
77
78 function error_msg() {
79     echo -e " $RED$1$DEFAULT";
80 }
81
82 # -----
83 # Input parameters tests.
84 # -----
85
86 # Define the expected outputs of each of the test cases with ↵
87 # its associated
88 # test functions.

```

```
84 EXPECTED_OUTPUT_INEXISTENT_INPUT_STREAM=("ERROR: Can't open ↵
    input stream.")
85
86 function test1_parameter_input_inexistent_stream(){
87     header "TEST1: inexistent 'input' stream."
88
89     commands=(
90         "-i 1"
91         "-i hola.bin"
92     )
93
94     for i in "${commands[@]}"
95     do
96
97         msg_testing "$PROGRAM_NAME $i"
98
99         PROGRAM_OUTPUT=$($PROGRAM_NAME $i 2>&1)
100
101         if [[ "$EXPECTED_OUTPUT_INEXISTENT_INPUT_STREAM" == "↵
            $PROGRAM_OUTPUT" ]]; then
102             msg_true "$PROGRAM_OUTPUT"
103         else
104             msg_false "$PROGRAM_OUTPUT"
105             failedTests=$((failedTests+1));
106         fi
107
108     done
109 }
110
111 EXPECTED_OUTPUT_INPUT_NO_ARGUMENT="(./tp0: option requires an ↵
    argument -- 'i')"
112
113 function test11_parameter_input_no_argument(){
114     header "TEST11: no 'input' option parameters."
115
116     commands=("-i ")
117
118     for i in "${commands[@]}"
119     do
120
121         msg_testing "$PROGRAM_NAME $i"
122
123         PROGRAM_OUTPUT=$($PROGRAM_NAME $i 2>&1)
124
125         if [[ "$EXPECTED_OUTPUT_INPUT_NO_ARGUMENT" == "↵
            $PROGRAM_OUTPUT" ]]; then
126             msg_true "$PROGRAM_OUTPUT"
```

```

127     else
128         msg_false "$PROGRAM_OUTPUT"
129         failedTests=$((failedTests+1));
130     fi
131
132     done
133 }
134
135 EXPECTED_OUTPUT_INPUT_INVALID_STREAM=("ERROR: Invalid input ↵
stream.")
136
137 function test12_parameter_input_invalid_stream(){
138     header "TEST12: invalid 'input' stream."
139
140     commands=(
141         "-i ."
142         "-i .."
143         "-i /"
144         "-i //"
145     )
146
147     for i in "${commands[@]}"; do
148
149
150         msg_testing "$PROGRAM_NAME $i"
151
152         PROGRAM_OUTPUT=$((PROGRAM_NAME $i 2>&1)
153
154         if [[ "$EXPECTED_OUTPUT_INPUT_INVALID_STREAM" == "↵
$PROGRAM_OUTPUT" ]]; then
155             msg_true "$PROGRAM_OUTPUT"
156         else
157             msg_false "$PROGRAM_OUTPUT"
158             failedTests=$((failedTests+1));
159         fi
160
161     done
162 }
163
164 EXPECTED_OUTPUT_INVALID_OUTPUT_STREAM=("ERROR: Invalid output ↵
stream.")
165
166 function test2_parameter_output_stream(){
167     header "TEST2: invalid 'output' stream."
168
169     commands=(
170         "-i ../tests/leviathan.input -o ."

```

```

171 "-i ../tests/leviathan.input -o .."
172 "-i ../tests/leviathan.input -o /"
173 "-i ../tests/leviathan.input -o //"
174 )
175
176 for i in "${commands[@]}"
177 do
178
179     msg_testing "$PROGRAM_NAME $i"
180
181     PROGRAM_OUTPUT=$($PROGRAM_NAME $i 2>&1)
182
183     if [[ "$EXPECTED_OUTPUT_INVALID_OUTPUT_STREAM" == "↵
184         $PROGRAM_OUTPUT" ]]; then
185         msg_true "$PROGRAM_OUTPUT"
186     else
187         msg_false "$PROGRAM_OUTPUT"
188         failedTests=$((failedTests+1));
189     fi
190 done
191 }
192
193 EXPECTED_OUTPUT_OUTPUT_NO_ARGUMENT="(./tp0: option requires an↵
194     argument -- 'o')"
195
196 function test21_parameter_output_no_argument(){
197     header "TEST21: no 'output' option parameters."
198
199     commands=("-o ")
200
201     for i in "${commands[@]}"
202     do
203
204         msg_testing "$PROGRAM_NAME $i"
205
206         PROGRAM_OUTPUT=$($PROGRAM_NAME $i 2>&1)
207
208         if [[ "$EXPECTED_OUTPUT_OUTPUT_NO_ARGUMENT" == "↵
209             $PROGRAM_OUTPUT" ]]; then
210             msg_true "$PROGRAM_OUTPUT"
211         else
212             msg_false "$PROGRAM_OUTPUT"
213             failedTests=$((failedTests+1));
214         fi
215     done

```



```
215 }
216
217 EXPECTED_OUTPUT_INVALID_ACTION=("ERROR: Invalid action ↵
    argument.")
218
219 function test3_parameter_action(){
220     header "TEST3: invalid 'action' parameters."
221
222     commands=(
223         "-a bad_action"
224         "-a 1"
225         "-a ."
226         "-a .."
227         "-a /"
228         "-a //"
229         "-a $"
230     )
231
232     for i in "${commands[@]}"
233     do
234
235         msg_testing "$PROGRAM_NAME $i"
236
237         PROGRAM_OUTPUT=$(($PROGRAM_NAME $i 2>&1))
238
239         if [[ "$EXPECTED_OUTPUT_INVALID_ACTION" == "↵
            $PROGRAM_OUTPUT" ]]; then
240             msg_true "$PROGRAM_OUTPUT"
241         else
242             msg_false "$PROGRAM_OUTPUT"
243             failedTests=$((failedTests+1));
244         fi
245
246     done
247 }
248
249 EXPECTED_OUTPUT_ACTION_NO_ARGUMENT="(./tp0: option requires an↵
    argument -- 'a')
250
251 function test31_parameter_action_no_argument(){
252     header "TEST31: no 'action' option parameters."
253
254     commands=("-a ")
255
256     for i in "${commands[@]}"
257     do
258
```

```

259     msg_testing "$PROGRAM_NAME $i"
260
261     PROGRAM_OUTPUT=$(($PROGRAM_NAME $i 2>&1)
262
263     if [[ "$EXPECTED_OUTPUT_ACTION_NO_ARGUMENT" == "↩"
264         $PROGRAM_OUTPUT" ]]; then
265         msg_true "$PROGRAM_OUTPUT"
266     else
267         msg_false "$PROGRAM_OUTPUT"
268         failedTests=$((failedTests+1));
269     fi
270 done
271 }
272
273 EXPECTED_OUTPUT_VALID_PARAMETERS=()
274
275 function test4_valid_parameters(){
276     header "TEST4: all options with correct parameters."
277
278     commands=(
279         "-a encode -i ../tests/test1.bin -o ../tests/test1_out.bin"
280         "-a decode -i ../tests/test1_out.bin -o ../tests/test2.bin")
281
282     for i in "${commands[@]}"
283     do
284
285         msg_testing "$PROGRAM_NAME $i"
286
287         PROGRAM_OUTPUT=$(($PROGRAM_NAME $i 2>&1)
288
289
290         if [[ "$EXPECTED_OUTPUT_VALID_PARAMETERS" == "↩"
291             $PROGRAM_OUTPUT" ]]; then
292             msg_true "$PROGRAM_OUTPUT"
293         else
294             msg_false "$PROGRAM_OUTPUT"
295             failedTests=$((failedTests+1));
296         fi
297     done
298 }
299
300 # -----
301 # Input-output validation tests.
302 # -----
303 function IO_validation_passed(){

```

```

304     echo -e "$GREEN\OPASSED: $DEFAULT $1"
305 }
306 function IO_validation_failed(){
307     echo -e "$RED\OFAILED $DEFAULT $1"
308 }
309
310 TESTS_DIR="./tests";
311 mkdir $TESTS_DIR;
312
313 function test5_IO_validation(){
314     header "TEST5: input-output should be the same."
315
316     n=1;
317     # nLimit=$((1024*1000));
318     nLimit=10;
319
320     while [ $n -le $nLimit ]
321     do
322         head -c $n </dev/urandom >$TESTS_DIR/in.bin;
323         # yes | head -c $n >$TESTS_DIR/in.bin;
324         $PROGRAM_NAME -a encode -i $TESTS_DIR/in.bin -o $TESTS_DIR/←
325         /out.b64;
326         $PROGRAM_NAME -a decode -i $TESTS_DIR/out.b64 -o ←
327         $TESTS_DIR/out.bin;
328
329         diff_result="$$(diff $TESTS_DIR/in.bin $TESTS_DIR/out.bin)"←
330         ;
331
332         if [[ -z ${diff_result} ]]; then ;;
333             IO_validation_passed "n = $n";
334         else
335             IO_validation_failed "n = $n";
336             error_msg "in.bin";
337             cat $TESTS_DIR/in.bin | od -A x -t x1z -v;
338             error_msg "out.b64";
339             cat $TESTS_DIR/out.b64 | od -A x -t x1z -v;
340             error_msg "out.bin";
341             cat $TESTS_DIR/out.bin | od -A x -t x1z -v;
342             failedTests=$((failedTests+1));
343             break;
344         fi
345
346         # n=$((n*2));
347         n=$((n+1));
348
349         rm -f $TESTS_DIR/in.bin $TESTS_DIR/out.b64 $TESTS_DIR/out.←
350         bin

```

```

347     done
348 }
349
350 function test51_IO_validation(){
351     header "TEST51: input known text with known encoding."
352
353     $PROGRAM_NAME -a encode -i $TESTS_DIR/leviathan.input -o ↵
        $TESTS_DIR/leviathan_out.b64;
354
355     diff_result="$(diff $TESTS_DIR/leviathan_out.b64 $TESTS_DIR/↵
        leviathan_out_truth.b64)";
356
357     if [[ -z ${diff_result} ]]; then ;;
358         IO_validation_passed "No differences.";
359     else
360         IO_validation_failed "Differences: \n${diff_result}";
361         failedTests=$((failedTests+1));
362     fi
363 }
364
365 function test52_IO_validation(){
366     header "TEST52: Encode letter 'M'."
367
368     program_output="$(echo -n M | $PROGRAM_NAME)";
369     correct_output="TQ==";
370     diff_result="$(diff <(echo "$program_output" ) <(echo "↵
        $correct_output"))";
371
372     if [[ -z ${diff_result} ]]; then ;;
373         IO_validation_passed "No differences.";
374     else
375         IO_validation_failed "Differences: \n${diff_result}";
376         failedTests=$((failedTests+1));
377     fi
378 }
379
380 function test53_IO_validation(){
381     header "TEST53: Encode 'Ma'."
382
383     program_output="$(echo -n Ma | $PROGRAM_NAME)";
384     correct_output="TWE=";
385     diff_result="$(diff <(echo "$program_output" ) <(echo "↵
        $correct_output"))";
386
387     if [[ -z ${diff_result} ]]; then ;;
388         IO_validation_passed "No differences.";
389     else

```

```

390     IO_validation_failed "Differences: \n${diff_result}";
391     failedTests=$((failedTests+1));
392 fi
393 }
394
395 function test54_IO_validation(){
396     header "TEST54: Encode 'Man'."
397
398     program_output="$(echo -n Man | $PROGRAM_NAME)";
399     correct_output="TWFu";
400     diff_result="$(diff <(echo "$program_output" ) <(echo "$correct_output"))";
401
402     if [[ -z ${diff_result} ]]; then ;;
403         IO_validation_passed "No differences.";
404     else
405         IO_validation_failed "Differences: \n${diff_result}";
406         failedTests=$((failedTests+1));
407     fi
408 }
409
410 function test55_IO_validation(){
411     header "TEST55: Encode and decode 'Man'."
412
413     program_output="$(echo Man | $PROGRAM_NAME | $PROGRAM_NAME -a decode)";
414     correct_output="Man";
415     diff_result="$(diff <(echo "$program_output" ) <(echo "$correct_output"))";
416
417     if [[ -z ${diff_result} ]]; then ;;
418         IO_validation_passed "No differences.";
419     else
420         IO_validation_failed "Differences: \n${diff_result}";
421         failedTests=$((failedTests+1));
422     fi
423 }
424
425 function test56_IO_validation(){
426     header "TEST56: Check bit by bit."
427
428     program_output="$(echo -E xyz | $PROGRAM_NAME | $PROGRAM_NAME -a decode | od -t c)";
429     correct_output="0000000  x  y  z  \n
430 0000004";
431     diff_result="$(diff <(echo -E "$program_output" ) <(echo -E "$correct_output"))";

```

```

432
433     if [[ -z ${diff_result} ]]; then ;;
434         IO_validation_passed "No differences.";
435     else
436         IO_validation_failed "Differences: \n${diff_result}";
437         failedTests=$((failedTests+1));
438     fi
439 }
440
441 function test57_IO_validation(){
442     header "TEST57: Check max line length and number of encoded ↵
         bytes."
443
444     program_output_line_count="$(echo -n "$(yes | head -c 1024 | ↵
         $PROGRAM_NAME -a encode)" | wc -l)";
445
446     # 1024 bytes[base256] => (8190+2) bits => 1365 bytes[base64]↵
         + 2 bits
447     # 1365 bytes[base64] + 2 bits + '==' => 1366 bytes[base64]
448     # floor(1366 bytes[base64] / 76 charEachLine) => 17 lines
449     correct_output_line_count="17";
450     diff_result_line_count="$(diff <(echo "↵
         $program_output_line_count" ) <(echo "↵
         $correct_output_line_count"))";
451
452     if [[ -z ${diff_result_line_count} ]]; then ;;
453         IO_validation_passed "No differences in line count.";
454     else
455         IO_validation_failed "Differences in line count:
456         Program output:${program_output_line_count}
457         Correct output:${correct_output_line_count}";
458         failedTests=$((failedTests+1));
459     fi
460
461     program_output_word_count="$(yes | head -c 1024 | ↵
         $PROGRAM_NAME -a encode | $PROGRAM_NAME -a decode | wc -c↵
         )";
462     correct_output_word_count="1024";
463     diff_result_word_count="$(diff <(echo "↵
         $program_output_word_count" ) <(echo "↵
         $correct_output_word_count"))";
464
465     if [[ -z ${diff_result_word_count} ]]; then ;;
466         IO_validation_passed "No differences in word count.";
467     else
468         IO_validation_failed "Differences in word count:
469         Program output:${program_output_word_count}

```

```

470     Correct output:${correct_output_word_count}";
471     failedTests=$((failedTests+1));
472 fi
473 }
474
475 # -----
476 # Encoding-Decoding execution times tests.
477 # -----
478 function test6_encoding_execution_times(){
479     header "TEST6: encoding execution times."
480
481     n=1;
482     nLimit=$((1024*10000));
483
484     while [ $n -le $nLimit ]
485     do
486         head -c $n </dev/urandom >$TESTS_DIR/in.bin;
487         ts=$(date +%s%N);
488         $PROGRAM_NAME -a encode -i $TESTS_DIR/in.bin -o $TESTS_DIR/
489         /out.b64;
490         tt=$((($date +%s%N) - $ts)/1000000);
491         printf 'n: %-10d %10s %.2f [ms]\n' "$n" " " "$tt"
492
493         n=$((n*2));
494
495         rm -f $TESTS_DIR/in.bin $TESTS_DIR/out.b64 $TESTS_DIR/out.
496         bin
497     done
498 }
499
500 function test7_decoding_execution_times(){
501     header "TEST7: decoding execution times."
502
503     n=1;
504     nLimit=$((1024*10000));
505
506     while [ $n -le $nLimit ]
507     do
508         head -c $n </dev/urandom >$TESTS_DIR/in.bin;
509         $PROGRAM_NAME -a encode -i $TESTS_DIR/in.bin -o $TESTS_DIR/
510         /out.b64;
511         ts=$(date +%s%N);
512         $PROGRAM_NAME -a decode -i $TESTS_DIR/out.b64 -o ↵
513         $TESTS_DIR/out.bin;
514         tt=$((($date +%s%N) - $ts)/1000000);
515         printf 'n: %-10d %10s %.2f [ms]\n' "$n" " " "$tt"
516     done
517 }

```

```
513     n=$((n*2));
514
515     rm -f $TESTS_DIR/in.bin $TESTS_DIR/out.b64 $TESTS_DIR/out.↵
        bin
516 done
517 }
518
519 # -----
520 # Run the tests.
521 # -----
522 test1_parameter_input_inexistent_stream
523 test11_parameter_input_no_argument
524 test12_parameter_input_invalid_stream
525 test2_parameter_output_stream
526 test21_parameter_output_no_argument
527 test3_parameter_action
528 test31_parameter_action_no_argument
529 test4_valid_parameters
530 test5_IO_validation
531 test51_IO_validation
532 test52_IO_validation
533 test53_IO_validation
534 test54_IO_validation
535 test55_IO_validation
536 test56_IO_validation
537 test57_IO_validation
538 test6_encoding_execution_times
539 test7_decoding_execution_times
540
541 header "Test suite ended."
542
543 if [[ $failedTests -eq $zero ]]; then
544     success_msg "All tests passed.";
545 else
546     error_msg "Failed tests: $failedTests";
547 fi
```


C. Código fuente

C.0.1. main.c

```
1  /* -----
2  @Title:   FIUBA - 66.20 Organizacion de Computadoras.
3  @Project: TP0 - Infraestructura basica.
4  -----
5  @Filename: main.c
6  -----
7  @Authors:
8      Husain, Ignacio Santiago.
9          santiago.husain at gmail dot com
10     Pesado, Lucia.
11         luupesado at gmail dot com
12     Verstraeten, Federico.
13         federico.verstraeten at gmail dot com
14
15 @Date:           07-Sep-2018 3:46:28 pm
16 @Last modified by: Ignacio Santiago Husain
17 @Last modified time: 17-Sep-2018 12:12:16 am
18
19 @Copyright(C):
20     This file is part of 'TP0 - Infraestructura basica.'.
21     Unauthorized copying or use of this file via any medium
22     is strictly prohibited.
23 -----
24
25 Check memory leaks with the following command:
26 valgrind --tool=memcheck --leak-check=full \
27 --show-leak-kinds=all -v ./tp0
28
29 ----- */
30 #include <stdio.h>
31 #include <stdlib.h>
32 #include <string.h>
33 #include <unistd.h>
34 #include "common.h"
35 #include "decoder.h"
36 #include "encoder.h"
37 #include "parser.h"
38
39 int main(int argc, char **argv)
40 {
41     params_t params;
42
```

```
43  /* Initialize memory block with zeroes.*/
44  memset(&params, 0, sizeof(params_t));
45
46  /* We parse the command line and check for errors. */
47  outputCode cmdParsingState = parseCmdline(argc, argv, &↵
      params);
48  if (cmdParsingState == outERROR)
49  {
50      exit(EXIT_FAILURE);
51  }
52
53  outputCode transformationState;
54  if (strcmp(params.action, ENCODE_STR_TOKEN) == 0)
55  {
56      transformationState = encode(&params);
57  }
58  else
59  {
60      transformationState = decode(&params);
61  }
62  if (transformationState == outERROR)
63  {
64      exit(EXIT_FAILURE);
65  }
66
67  /* Close and free what is left. */
68  fclose(params.inputStream);
69  fclose(params.outputStream);
70
71  return EXIT_SUCCESS;
72 }
```

C.0.2. common.h

```

1  /* -----
2  @Title:   FIUBA - 66.20 Organizacion de Computadoras.
3  @Project: TP0 - Infraestructura basica.
4  -----
5  @Filename: common.h
6  -----
7  @Authors:
8      Husain, Ignacio Santiago.
9          santiago.husain at gmail dot com
10     Pesado, Lucia.
11         luupesado at gmail dot com
12     Verstraeten, Federico.
13         federico.verstraeten at gmail dot com
14
15 @Date:           12-Sep-2018 11:36:31 am
16 @Last modified by: Ignacio Santiago Husain
17 @Last modified time: 18-Sep-2018 3:28:10 pm
18
19 @Copyright(C):
20     This file is part of 'TP0 - Infraestructura basica.'.
21     Unauthorized copying or use of this file via any medium
22     is strictly prohibited.
23 -----
24
25 Common program structures.
26
27 ----- */
28 #ifndef COMMON__H
29 #define COMMON__H
30
31 #ifndef VERSION
32 #define VERSION "1.0.0"
33 #endif
34
35 #define SIZETABLEB64 64
36
37 static const char translationTableB64[SIZETABLEB64] =
38     "←
39         ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789←
40     +/";
41
42 typedef struct params_t
43 {
44     char *action;

```

```
43     FILE *inputStream;  
44     FILE *outputStream;  
45 } params_t;  
46  
47 typedef enum outputCodes_ { outOK, outERROR } outputCode;  
48  
49 #endif
```

C.0.3. decoder.h

```

1  /* -----
2  @Title:   FIUBA - 66.20 Organizacion de Computadoras.
3  @Project: TP0 - Infraestructura basica.
4  -----
5  @Filename: decoder.h
6  -----
7  @Authors:
8      Husain, Ignacio Santiago.
9          santiago.husain at gmail dot com
10     Pesado, Lucia.
11         luupesado at gmail dot com
12     Verstraeten, Federico.
13         federico.verstraeten at gmail dot com
14
15 @Date:           12-Sep-2018 11:53:31 am
16 @Last modified by: Ignacio Santiago Husain
17 @Last modified time: 18-Sep-2018 3:28:24 pm
18
19 @Copyright(C):
20     This file is part of 'TP0 - Infraestructura basica.'.
21     Unauthorized copying or use of this file via any medium
22     is strictly prohibited.
23 -----
24
25 PUT DESCRIPTION HERE.
26
27 ----- */
28 #ifndef DECODER__H
29 #define DECODER__H
30
31 #include <stdio.h>
32 #include <string.h>
33 #include "common.h"
34 #include "messages.h"
35
36 #define BYTE_DEC_MASK 0xFF000000
37 #define SIZEINDEX 4
38 #define PADDING_DEC '='
39 #define PADD_INDEX 0
40 #define OUTPUT_BLOCK_SIZE 3
41 #define BIT_PER_BYTE 8
42
43 outputCode base64ToBase256(unsigned char outChar[], unsigned char inChar[]);

```

```
44  outputCode decode(params_t *params);  
45  
46  #endif
```

C.0.4. decoder.c

```

1  /* -----
2  @Title:   FIUBA - 66.20 Organizacion de Computadoras.
3  @Project: TP0 - Infraestructura basica.
4  -----
5  @Filename: decoder.c
6  -----
7  @Authors:
8      Husain, Ignacio Santiago.
9          santiago.husain at gmail dot com
10     Pesado, Lucia.
11         luupesado at gmail dot com
12     Verstraeten, Federico.
13         federico.verstraeten at gmail dot com
14
15 @Date:           12-Sep-2018 11:21:30 am
16 @Last modified by: Ignacio Santiago Husain
17 @Last modified time: 18-Sep-2018 3:28:13 pm
18
19 @Copyright(C):
20     This file is part of 'TP0 - Infraestructura basica.'.
21     Unauthorized copying or use of this file via any medium
22     is strictly prohibited.
23 -----
24
25 PUT DESCRIPTION HERE.
26
27 ----- */
28 #include "decoder.h"
29
30 outputCode base64ToBase256(unsigned char outChar[], unsigned ↵
31     char inChar[])
32 {
33     unsigned int bitMask = BYTE_DEC_MASK;
34     size_t index1 = 0;
35     size_t index2 = 0;
36     unsigned char indexTable[SIZEINDEX] = {0, 0, 0, 0};
37     unsigned int charHolder = 0;
38     unsigned int bitPattern = 0;
39     unsigned char accumBit = 0;
40
41     /* Search char index in translationTableB64 */
42     for (index1 = 0; index1 < SIZEINDEX; index1++)
43     {
44         if (inChar[index1] == PADDING_DEC)

```

```

44     {
45         indexTable[index1] = PADD_INDEX;
46         continue;
47     }
48
49     for (index2 = 0; index2 < SIZETABLEB64; ++index2)
50     {
51         if (inChar[index1] == translationTableB64[index2])
52         {
53             indexTable[index1] = index2;
54             break;
55         }
56     }
57     if (index2 >= SIZETABLEB64)
58     {
59         fprintf(stderr, "ERROR: Character is not in Base64 Table↵
60             \n");
61         return outERROR;
62     }
63
64     for (index1 = 0; index1 < SIZEINDEX; index1++)
65     {
66         accumBit += 2;
67         charHolder = (unsigned int)indexTable[index1];
68         charHolder <<=
69             (SIZEINDEX - 1 - index1) * sizeof(unsigned char) * ↵
70             BIT_PER_BYTE;
71         charHolder <<= accumBit;
72         bitPattern = (bitPattern | charHolder);
73     }
74
75     index1 = 0;
76     do
77     {
78         /* Extract the decoded character from bitPattern */
79         charHolder = (bitPattern & bitMask);
80
81         /* Shift right the decoded character to the correct ↵
82            position. */
83         charHolder >>=
84             (SIZEINDEX - 1 - index1) * sizeof(unsigned char) * ↵
85             BIT_PER_BYTE;
86
87         /* Store in outChar */
88         outChar[index1] = (unsigned char)charHolder;

```



```

87     /* Shift right 0,8,16...bits the bitMask */
88     bitMask >>= sizeof(unsigned char) * BIT_PER_BYTE;
89     index1++;
90 } while (charHolder != 0 && (index1 < SIZEINDEX));
91
92 return outOK;
93 }
94
95 outputCode decode(params_t *params)
96 {
97     unsigned char readChar;
98     unsigned char inChar[SIZEINDEX] = {};
99     unsigned char outChar[OUTPUT_BLOCK_SIZE] = {};
100    unsigned char index1 = 0;
101
102    while (1)
103    {
104        /* Load input buffer inChar */
105        for (index1 = 0; index1 < SIZEINDEX; ++index1)
106        {
107            readChar = getc(params->inputStream);
108            if (ferror(params->inputStream))
109            {
110                fprintf(stderr, ERROR_INPUT_STREAM_READING_MSG);
111                return outERROR;
112            }
113
114            /* Discards WhiteSpaces detected */
115            if (readChar == '\n' || readChar == '\t' || readChar == ' ')
116            {
117                index1--;
118                continue;
119            }
120            else
121            {
122                inChar[index1] = readChar;
123            }
124
125            /* EOF */
126            if (feof(params->inputStream))
127            {
128                /* If there are still chars in the buffer, we flush it */
129                . /*
130                if (index1 != 0)
131                {
132                    if (base64ToBase256(outChar, inChar) == outERROR)

```

```
132     {
133         return outERROR;
134     }
135
136     fputc(outChar, params->outputStream);
137     if (ferror(params->outputStream))
138     {
139         fprintf(stderr, ERROR_OUTPUT_STREAM_WRITING_MSG);
140         return outERROR;
141     }
142 }
143 return outOK;
144 }
145 }
146
147 /* Translate inChar into base256 */
148 if (base64ToBase256(outChar, inChar) == outERROR)
149 {
150     return outERROR;
151 }
152
153 fputc(outChar, params->outputStream);
154 if (ferror(params->outputStream))
155 {
156     fprintf(stderr, ERROR_OUTPUT_STREAM_WRITING_MSG);
157     return outERROR;
158 }
159 }
160
161 return outERROR;
162 }
```

C.0.5. encoder.h

```

1  /* -----
2  @Title:   FIUBA - 66.20 Organizacion de Computadoras.
3  @Project: TP0 - Infraestructura basica.
4  -----
5  @Filename: encoder.h
6  -----
7  @Authors:
8      Husain, Ignacio Santiago.
9          santiago.husain at gmail dot com
10     Pesado, Lucia.
11         luupesado at gmail dot com
12     Verstraeten, Federico.
13         federico.verstraeten at gmail dot com
14
15 @Date:           12-Sep-2018 11:52:58 am
16 @Last modified by: Ignacio Santiago Husain
17 @Last modified time: 18-Sep-2018 3:28:31 pm
18
19 @Copyright(C):
20     This file is part of 'TP0 - Infraestructura basica.'.
21     Unauthorized copying or use of this file via any medium
22     is strictly prohibited.
23 -----
24
25 PUT DESCRIPTION HERE.
26
27 ----- */
28 #ifndef ENCODER__H
29 #define ENCODER__H
30
31 #include <stdio.h>
32 #include <string.h>
33 #include "common.h"
34 #include "messages.h"
35
36 #define BYTE_ENC_MASK 0xFC
37 #define BYTE_ZEROS 0x00
38 #define MAX6BIT 6
39 #define PADDING "="
40 #define MAX_OUT_BUFFER 5
41 #define MAX_LINE LENGHT 76
42
43 unsigned char base256ToBase64(char *outChar, const char inChar↵
    );

```

```
44  outputCode  encode(params_t *params);  
45  
46  #endif
```

C.0.6. encoder.c

```

1  /* -----
2  @Title:   FIUBA - 66.20 Organizacion de Computadoras.
3  @Project: TP0 - Infraestructura basica.
4  -----
5  @Filename: encoder.c
6  -----
7  @Authors:
8      Husain, Ignacio Santiago.
9          santiago.husain at gmail dot com
10     Pesado, Lucia.
11         luupesado at gmail dot com
12     Verstraeten, Federico.
13         federico.verstraeten at gmail dot com
14
15 @Date:           12-Sep-2018 11:21:26 am
16 @Last modified by: Ignacio Santiago Husain
17 @Last modified time: 18-Sep-2018 6:43:30 pm
18
19 @Copyright(C):
20     This file is part of 'TP0 - Infraestructura basica.'.
21     Unauthorized copying or use of this file via any medium
22     is strictly prohibited.
23 -----
24
25 PUT DESCRIPTION HERE.
26
27 ----- */
28 #include "encoder.h"
29
30 unsigned char base256ToBase64(char *outChar, const char inChar←
31     )
32 {
33     unsigned char headByte = BYTE_ZEROS;
34     unsigned char prevByte = BYTE_ZEROS;
35     static unsigned char tailByte = BYTE_ZEROS;
36     static unsigned char bitMask = BYTE_ENC_MASK;
37     static unsigned int shiftRightBit = 2;
38     unsigned char encodedCharsCount = 0;
39
40     /* Backup the previous tailByte*/
41     prevByte = tailByte;
42
43     /*Padding: The last encoded block contain less 6bit*/
44     if ((inChar == EOF))

```

```
44 {
45     if (shiftRightBit == 6)
46     {
47         headByte = (prevByte | BYTE_ZEROS);
48         strncpy(outChar, &translationTableB64[headByte], 1);
49         strncat(outChar, PADDING, 1);
50         return (encodedCharsCount + 2);
51     }
52     else if (shiftRightBit == 4)
53     {
54         headByte = (prevByte | BYTE_ZEROS);
55         strncpy(outChar, &translationTableB64[headByte], 1);
56         strncat(outChar, PADDING, 1);
57         strncat(outChar, PADDING, 1);
58         return (encodedCharsCount + 3);
59     }
60     else
61         return encodedCharsCount;
62 }
63
64 /* Save the head of input char*/
65 headByte = inChar & bitMask;
66
67 /* Shift right 2, 4 or 6bit the headByte*/
68 headByte >>= shiftRightBit;
69
70 /* Save the tail input char*/
71 tailByte = inChar & (~bitMask);
72
73 /* Shift left 4, 2 or 0bit the tailByte*/
74 tailByte <<= (MAX6BIT - shiftRightBit);
75
76 /* Merge previous tailByte and current headByte*/
77 headByte = (prevByte | headByte);
78
79 /*Print translation in outChar*/
80 strncpy(outChar, &translationTableB64[headByte], 1);
81 encodedCharsCount++;
82
83 shiftRightBit += 2;
84
85 /* Shift left 2 bits the mask */
86 if (!(bitMask <<= 2))
87 {
88     /* Restart mask at the beginning */
89     bitMask = BYTE_ENC_MASK;
90     shiftRightBit = 2;
```

```
91
92     /* Print tailByte and clear*/
93     strncat(outChar, &translationTableB64[tailByte], 1);
94     encodedCharsCount++;
95
96     tailByte = BYTE_ZEROS;
97 }
98
99     return encodedCharsCount;
100 }
101
102 outputCode encode(params_t *params)
103 {
104     /* TODO:  revisar si estos char pueden o deben ser unsigned.↵
105        */
106     char inChar;
107     char outChar[MAX_OUT_BUFFER] = {};
108     unsigned char totalEncodedCharsCount = 0, encodedCharsCount ↵
109         = 0;
110
111     do
112     {
113         /* Clear outChar. */
114         memset(outChar, 0, sizeof(outChar));
115         inChar = getc(params->inputStream);
116         if (ferror(params->inputStream))
117         {
118             fprintf(stderr, ERROR_INPUT_STREAM_READING_MSG);
119             return outERROR;
120         }
121         encodedCharsCount = base256ToBase64(outChar, inChar);
122
123         if ((totalEncodedCharsCount + encodedCharsCount) <= ↵
124             MAX_LINE_LENGTH)
125         {
126             totalEncodedCharsCount += encodedCharsCount;
127         }
128         else
129         {
130             fputs("\n", params->outputStream);
131             if (ferror(params->outputStream))
132             {
133                 fprintf(stderr, ERROR_OUTPUT_STREAM_WRITING_MSG);
134                 return outERROR;
135             }
136             totalEncodedCharsCount = encodedCharsCount;
137         }
138     }
```

```
135
136     fputs(outChar, params->outputStream);
137     if (ferror(params->outputStream))
138     {
139         fprintf(stderr, ERROR_OUTPUT_STREAM_WRITING_MSG);
140         return outERROR;
141     }
142
143 } while (inChar != EOF);
144
145 return outOK;
146 }
```


C.0.7. parser.h

```

1  /* -----
2  @Title:   FIUBA - 66.20 Organizacion de Computadoras.
3  @Project: TP0 - Infraestructura basica.
4  -----
5  @Filename: parser.h
6  -----
7  @Authors:
8      Husain, Ignacio Santiago.
9          santiago.husain at gmail dot com
10     Pesado, Lucia.
11         luupesado at gmail dot com
12     Verstraeten, Federico.
13         federico.verstraeten at gmail dot com
14
15 @Date:           12-Sep-2018 11:50:53 am
16 @Last modified by: Ignacio Santiago Husain
17 @Last modified time: 12-Sep-2018 12:39:22 pm
18
19 @Copyright(C):
20     This file is part of 'TP0 - Infraestructura basica.'.
21     Unauthorized copying or use of this file via any medium
22     is strictly prohibited.
23 -----
24
25 PUT DESCRIPTION HERE.
26
27 ----- */
28 #ifndef PARSER__H
29 #define PARSER__H
30
31 #include <getopt.h>
32 #include <stdio.h>
33 #include <stdlib.h>
34 #include <string.h>
35 #include "common.h"
36 #include "messages.h"
37
38 #ifndef STD_STREAM_TOKEN
39 #define STD_STREAM_TOKEN "-"
40 #endif
41
42 /* Actions definitions */
43 #ifndef ENCODE_STR_TOKEN
44 #define ENCODE_STR_TOKEN "encode"

```

```
45 #endif
46 #ifndef DECODE_STR_TOKEN
47 #define DECODE_STR_TOKEN "decode"
48 #endif
49
50 extern struct option cmdOptions[];
51
52 outputCode parseCmdline(int argc, char **argv, params_t *↵
    params);
53 outputCode optAction(char *arg, params_t *params);
54 outputCode optOutput(char *arg, params_t *params);
55 outputCode optInput(char *arg, params_t *params);
56 void optHelp(char *arg);
57 void optVersion(void);
58
59 #endif
```

C.0.8. parser.c

```

1  /* -----
2  @Title:   FIUBA - 66.20 Organizacion de Computadoras.
3  @Project: TP0 - Infraestructura basica.
4  -----
5  @Filename: parser.c
6  -----
7  @Authors:
8      Husain, Ignacio Santiago.
9          santiago.husain at gmail dot com
10     Pesado, Lucia.
11         luupesado at gmail dot com
12     Verstraeten, Federico.
13         federico.verstraeten at gmail dot com
14
15 @Date:           12-Sep-2018 11:21:22 am
16 @Last modified by: Ignacio Santiago Husain
17 @Last modified time: 18-Sep-2018 3:28:38 pm
18
19 @Copyright(C):
20     This file is part of 'TP0 - Infraestructura basica.'.
21     Unauthorized copying or use of this file via any medium
22     is strictly prohibited.
23 -----
24
25 PUT DESCRIPTION HERE.
26
27 ----- */
28 #include "parser.h"
29
30 /* The options are distinguished by the ASCII code of the
31  * 'char' variables. */
32 struct option cmdOptions[] = {"version", no_argument, NULL, '↵
33     V'},
34                                {"help", no_argument, NULL, 'h'↵
35                                },
36                                {"input", required_argument, ↵
37                                NULL, 'i'},
38                                {"output", required_argument, ↵
39                                NULL, 'o'},
40                                {"action", required_argument, ↵
41                                NULL, 'a'},
42                                {0, 0, 0, 0}};
43
44 void optVersion(void)

```

```

40 {
41     fprintf(stderr, "%s\n", VERSION);
42
43     exit(EXIT_SUCCESS);
44 }
45
46 void optHelp(char *arg)
47 {
48     if (arg == NULL)
49     {
50         fprintf(stderr, ERROR_ACTION_INVALID_ARGUMENT);
51         exit(EXIT_FAILURE);
52     }
53     fprintf(stderr, "Usage:\n");
54     fprintf(stderr, "  %s -h\n", arg);
55     fprintf(stderr, "  %s -V\n", arg);
56     fprintf(stderr, "  %s [options]\n", arg);
57     fprintf(stderr, "Options:\n");
58     fprintf(stderr, "-V, --version\tPrint version and quit.\n");
59     fprintf(stderr, "-i, --input\tLocation of the input file.\n");
60     fprintf(stderr, "-o, --output\tLocation of the output file.\n");
61     fprintf(stderr,
62         "-a, --action\tProgram action: encode (default) or \n");
63     fprintf(stderr, "Examples:\n");
64     fprintf(stderr, "  %s -a encode -i ~/input -o ~/output\n", arg);
65     fprintf(stderr, "  %s -a decode\n", arg);
66
67     exit(EXIT_SUCCESS);
68 }
69
70 outputCode validateStreamName(char *streamName)
71 {
72     if (streamName == NULL)
73     {
74         return outERROR;
75     }
76
77     if (!strcmp(streamName, ".") || !strcmp(streamName, "..") ||
78         !strcmp(streamName, "/") || !strcmp(streamName, "//"))
79     {
80         return outERROR;
81     }
82

```

```
83     return outOK;
84 }
85
86 outputCode optInput(char *arg, params_t *params)
87 {
88     if (validateStreamName(arg) == outERROR)
89     {
90         fprintf(stderr, ERROR_INVALID_INPUT_STREAM);
91         return outERROR;
92     }
93
94     if (strcmp(arg, STD_STREAM_TOKEN) == 0)
95     {
96         params->inputStream = stdin;
97     }
98     else
99     {
100         params->inputStream = fopen(arg, "rb");
101     }
102
103     if ((params->inputStream) == NULL)
104     {
105         fprintf(stderr, ERROR_OPENING_INPUT_STREAM);
106         return outERROR;
107     }
108
109     return outOK;
110 }
111
112 outputCode optOutput(char *arg, params_t *params)
113 {
114     if (validateStreamName(arg) == outERROR)
115     {
116         fprintf(stderr, ERROR_INVALID_OUTPUT_STREAM);
117         return outERROR;
118     }
119
120     if (strcmp(arg, STD_STREAM_TOKEN) == 0)
121     {
122         params->outputStream = stdout;
123     }
124     else
125     {
126         params->outputStream = fopen(arg, "wb");
127     }
128
129     if ((params->outputStream) == NULL)
```

```
130 {
131     fprintf(stderr, ERROR_OPENING_OUTPUT_STREAM);
132     return outERROR;
133 }
134
135 return outOK;
136 }
137
138 outputCode optAction(char *arg, params_t *params)
139 {
140     if (arg == NULL)
141     {
142         fprintf(stderr, ERROR_ACTION_INVALID_ARGUMENT);
143         return outERROR;
144     }
145
146     if (strcmp(arg, ENCODE_STR_TOKEN) == 0)
147     {
148         params->action = ENCODE_STR_TOKEN;
149     }
150     else if (strcmp(arg, DECODE_STR_TOKEN) == 0)
151     {
152         params->action = DECODE_STR_TOKEN;
153     }
154     else
155     {
156         fprintf(stderr, ERROR_ACTION_INVALID_ARGUMENT);
157         return outERROR;
158     }
159
160     return outOK;
161 }
162
163 outputCode parseCmdline(int argc, char **argv, params_t *↵
    params)
164 {
165     int indexptr = 0;
166     int optCode;
167
168     outputCode optOutCode = outERROR;
169     char *programName = argv[0];
170
171     /* Set the default values. */
172     params->action = ENCODE_STR_TOKEN;
173     params->inputStream = stdin;
174     params->outputStream = stdout;
175
```

```
176  /* 'version' and 'help' have no arguments. The rest, do
177  * have, and are mandatory.*/
178  char *shortOpts = "Vhi:o:a:";
179
180  while ((optCode =
181          getopt_long(argc, argv, shortOpts, cmdOptions, &↵
                      indexptr)) != -1)
182  {
183      switch (optCode)
184      {
185          case 'V':
186              optVersion();
187              break;
188          case 'h':
189              optHelp(programName);
190              break;
191          case 'i':
192              optOutCode = optInput(optarg, params);
193              break;
194          case 'o':
195              optOutCode = optOutput(optarg, params);
196              break;
197          case 'a':
198              optOutCode = optAction(optarg, params);
199              break;
200          case '?':
201              /* getopt_long already printed an error message. */
202              optOutCode = outERROR;
203              break;
204      }
205      if (optOutCode == outERROR)
206      {
207          return outERROR;
208      }
209  }
210
211  return outOK;
212 }
```

C.0.9. messages.h

```

1  /* -----
2  @Title:   FIUBA - 66.20 Organizacion de Computadoras.
3  @Project: TP0 - Infraestructura basica.
4  -----
5  @Filename: messages.h
6  -----
7  @Authors:
8      Husain, Ignacio Santiago.
9          santiago.husain at gmail dot com
10     Pesado, Lucia.
11         luupesado at gmail dot com
12     Verstraeten, Federico.
13         federico.verstraeten at gmail dot com
14
15 @Date:           12-Sep-2018 11:32:42 am
16 @Last modified by: Ignacio Santiago Husain
17 @Last modified time: 12-Sep-2018 11:34:15 am
18
19 @Copyright(C):
20     This file is part of 'TP0 - Infraestructura basica.'.
21     Unauthorized copying or use of this file via any medium
22     is strictly prohibited.
23 -----
24
25 PUT DESCRIPTION HERE.
26
27 ----- */
28 #ifndef MESSAGES__H
29 #define MESSAGES__H
30
31 #ifndef ERROR_INVALID_INPUT_STREAM
32 #define ERROR_INVALID_INPUT_STREAM "ERROR: Invalid input ↵
33     stream.\n"
34 #endif
35 #ifndef ERROR_OPENING_INPUT_STREAM
36 #define ERROR_OPENING_INPUT_STREAM "ERROR: Can't open input ↵
37     stream.\n"
38 #endif
39 #ifndef ERROR_INVALID_OUTPUT_STREAM
40 #define ERROR_INVALID_OUTPUT_STREAM "ERROR: Invalid output ↵
41     stream.\n"
42 #endif
43 #ifndef ERROR_OPENING_OUTPUT_STREAM

```



```
41 #define ERROR_OPENING_OUTPUT_STREAM "ERROR: Can't open output ↵  
    stream.\n"  
42 #endif  
43 #ifndef ERROR_ACTION_INVALID_ARGUMENT  
44 #define ERROR_ACTION_INVALID_ARGUMENT "ERROR: Invalid action ↵  
    argument.\n"  
45 #endif  
46 #ifndef ERROR_OUTPUT_STREAM_WRITING_MSG  
47 #define ERROR_OUTPUT_STREAM_WRITING_MSG "Output error when ↵  
    writing stream.\n"  
48 #endif  
49 #ifndef ERROR_INPUT_STREAM_READING_MSG  
50 #define ERROR_INPUT_STREAM_READING_MSG "Input error when ↵  
    reading stream.\n"  
51 #endif  
52  
53 #endif
```

C.0.10. main.s

```

1  .file    1  "main.c"
2  .section .mdebug.abi32
3  .previous
4  .abicalls
5  .rdata
6  .align   2
7  .type    translationTableB64, @object
8  .size    translationTableB64, 64
9  translationTableB64:
10  .ascii   "↵
           ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123↵
           "
11  .ascii   "456789+/"
12  .align   2
13 $LC0:
14  .ascii   "encode\000"
15  .text
16  .align   2
17  .globl   main
18  .ent     main
19 main:
20  .frame    $fp,64,$31      # vars= 24, regs= 3/0, args= 16, ↵
           extra= 8
21  .mask     0xd0000000,-8
22  .fmask    0x00000000,0
23  .set      noreorder
24  .cpload   $25
25  .set      reorder
26  subu      $sp,$sp,64
27  .cprestore 16
28  sw        $31,56($sp)
29  sw        $fp,52($sp)
30  sw        $28,48($sp)
31  move      $fp,$sp
32  sw        $4,64($fp)
33  sw        $5,68($fp)
34  addu      $4,$fp,24
35  move      $5,$0
36  li        $6,12          # 0xc
37  la        $25,memset
38  jal       $31,$25
39  lw        $4,64($fp)
40  lw        $5,68($fp)
41  addu      $6,$fp,24

```

```
42     la    $25,parseCmdline
43     jal   $31,$25
44     sw    $2,40($fp)
45     lw    $3,40($fp)
46     li    $2,1                # 0x1
47     bne   $3,$2,$L18
48     li    $4,1                # 0x1
49     la    $25,exit
50     jal   $31,$25
51 $L18:
52     lw    $4,24($fp)
53     la    $5,$LC0
54     la    $25,strcmp
55     jal   $31,$25
56     bne   $2,$0,$L19
57     addu   $4,$fp,24
58     la    $25,encode
59     jal   $31,$25
60     sw    $2,44($fp)
61     b     $L20
62 $L19:
63     addu   $4,$fp,24
64     la    $25,decode
65     jal   $31,$25
66     sw    $2,44($fp)
67 $L20:
68     lw    $3,44($fp)
69     li    $2,1                # 0x1
70     bne   $3,$2,$L21
71     li    $4,1                # 0x1
72     la    $25,exit
73     jal   $31,$25
74 $L21:
75     lw    $4,28($fp)
76     la    $25,fclose
77     jal   $31,$25
78     lw    $4,32($fp)
79     la    $25,fclose
80     jal   $31,$25
81     move   $2,$0
82     move   $sp,$fp
83     lw    $31,56($sp)
84     lw    $fp,52($sp)
85     addu   $sp,$sp,64
86     j     $31
87     .end   main
88     .size  main,.-main
```

```
89 .ident "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520) "
```

C.0.11. parser.s

```

1      .file      1  "parser.c"
2      .section   .mdebug.abi32
3      .previous
4      .abicalls
5      .rdata
6      .align     2
7      .type      translationTableB64, @object
8      .size      translationTableB64, 64
9  translationTableB64:
10     .ascii      "↵
                     ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123↵
                     "
11     .ascii      "456789+/"
12     .align     2
13 $LC0:
14     .ascii      "version\000"
15     .align     2
16 $LC1:
17     .ascii      "help\000"
18     .align     2
19 $LC2:
20     .ascii      "input\000"
21     .align     2
22 $LC3:
23     .ascii      "output\000"
24     .align     2
25 $LC4:
26     .ascii      "action\000"
27     .globl      cmdOptions
28     .data
29     .align     2
30     .type      cmdOptions, @object
31     .size      cmdOptions, 96
32 cmdOptions:
33     .word      $LC0
34     .word      0
35     .word      0
36     .word      86
37     .word      $LC1
38     .word      0
39     .word      0
40     .word      104
41     .word      $LC2
42     .word      1

```

```
43     .word    0
44     .word    105
45     .word    $LC3
46     .word    1
47     .word    0
48     .word    111
49     .word    $LC4
50     .word    1
51     .word    0
52     .word    97
53     .word    0
54     .word    0
55     .word    0
56     .word    0
57     .rdata
58     .align   2
59 $LC5:
60     .ascii   "%s\n\000"
61     .align   2
62 $LC6:
63     .ascii   "1.0.0\000"
64     .text
65     .align   2
66     .globl   optVersion
67     .ent     optVersion
68 optVersion:
69     .frame   $fp,40,$31      # vars= 0, regs= 3/0, args= 16, ←
70         extra= 8
71     .mask    0xd0000000,-8
72     .fmask   0x00000000,0
73     .set     noreorder
74     .cpload  $25
75     .set     reorder
76     subu     $sp,$sp,40
77     .cpstore 16
78     sw       $31,32($sp)
79     sw       $fp,28($sp)
80     sw       $28,24($sp)
81     move     $fp,$sp
82     la       $4, __sF+176
83     la       $5,$LC5
84     la       $6,$LC6
85     la       $25,fprintf
86     jal      $31,$25
87     move     $4,$0
88     la       $25,exit
89     jal      $31,$25
```

```

89     .end      optVersion
90     .size     optVersion, .-optVersion
91     .rdata
92     .align    2
93 $LC7:
94     .ascii    "ERROR: Invalid action argument.\n\000"
95     .align    2
96 $LC8:
97     .ascii    "Usage:\n\000"
98     .align    2
99 $LC9:
100    .ascii    "  %s -h\n\000"
101    .align    2
102 $LC10:
103    .ascii    "  %s -V\n\000"
104    .align    2
105 $LC11:
106    .ascii    "  %s [options]\n\000"
107    .align    2
108 $LC12:
109    .ascii    "Options:\n\000"
110    .align    2
111 $LC13:
112    .ascii    "-V, --version\tPrint version and quit.\n\000"
113    .align    2
114 $LC14:
115    .ascii    "-i, --input\tLocation of the input file.\n\000"
116    .align    2
117 $LC15:
118    .ascii    "-o, --output\tLocation of the output file.\n\000"
119    .align    2
120 $LC16:
121    .ascii    "-a, --action\tProgram action: encode (default) or↵
122              decode"
123    .ascii    ".\n\000"
124    .align    2
125 $LC17:
126    .ascii    "Examples:\n\000"
127    .align    2
128 $LC18:
129    .ascii    "  %s -a encode -i ~/input -o ~/output\n\000"
130    .align    2
131 $LC19:
132    .ascii    "  %s -a decode\n\000"
133    .text
134    .align    2
135    .globl    optHelp

```

```

135     .ent      optHelp
136 optHelp:
137     .frame    $fp,40,$31      # vars= 0, regs= 3/0, args= 16, ↵
138         extra= 8
139     .mask     0xd0000000,-8
140     .fmask    0x00000000,0
141     .set      noreorder
142     .cpload   $25
143     .set      reorder
144     subu      $sp,$sp,40
145     .cprestore 16
146     sw        $31,32($sp)
147     sw        $fp,28($sp)
148     sw        $28,24($sp)
149     move      $fp,$sp
150     sw        $4,40($fp)
151     lw        $2,40($fp)
152     bne       $2,$0,$L19
153     la        $4, __sF+176
154     la        $5,$LC7
155     la        $25,fprintf
156     jal       $31,$25
157     li        $4,1            # 0x1
158     la        $25,exit
159     jal       $31,$25
160 $L19:
161     la        $4, __sF+176
162     la        $5,$LC8
163     la        $25,fprintf
164     jal       $31,$25
165     la        $4, __sF+176
166     la        $5,$LC9
167     lw        $6,40($fp)
168     la        $25,fprintf
169     jal       $31,$25
170     la        $4, __sF+176
171     la        $5,$LC10
172     lw        $6,40($fp)
173     la        $25,fprintf
174     jal       $31,$25
175     la        $4, __sF+176
176     la        $5,$LC11
177     lw        $6,40($fp)
178     la        $25,fprintf
179     jal       $31,$25
180     la        $4, __sF+176
181     la        $5,$LC12

```



```
181     la    $25,fprintf
182     jal   $31,$25
183     la    $4, __sF+176
184     la    $5,$LC13
185     la    $25,fprintf
186     jal   $31,$25
187     la    $4, __sF+176
188     la    $5,$LC14
189     la    $25,fprintf
190     jal   $31,$25
191     la    $4, __sF+176
192     la    $5,$LC15
193     la    $25,fprintf
194     jal   $31,$25
195     la    $4, __sF+176
196     la    $5,$LC16
197     la    $25,fprintf
198     jal   $31,$25
199     la    $4, __sF+176
200     la    $5,$LC17
201     la    $25,fprintf
202     jal   $31,$25
203     la    $4, __sF+176
204     la    $5,$LC18
205     lw    $6,40($fp)
206     la    $25,fprintf
207     jal   $31,$25
208     la    $4, __sF+176
209     la    $5,$LC19
210     lw    $6,40($fp)
211     la    $25,fprintf
212     jal   $31,$25
213     move   $4,$0
214     la    $25,exit
215     jal   $31,$25
216     .end    optHelp
217     .size   optHelp, .-optHelp
218     .rdata
219     .align  2
220 $LC20:
221     .ascii  ".\000"
222     .align  2
223 $LC21:
224     .ascii  "..\000"
225     .align  2
226 $LC22:
227     .ascii  " /\000"
```

```

228     .align    2
229 $LC23:
230     .ascii    "//\000"
231     .text
232     .align    2
233     .globl    validateStreamName
234     .ent      validateStreamName
235 validateStreamName:
236     .frame    $fp,48,$31          # vars= 8, regs= 3/0, args= 16, ←
        extra= 8
237     .mask     0xd0000000,-8
238     .fmask    0x00000000,0
239     .set      noreorder
240     .cpload   $25
241     .set      reorder
242     subu      $sp,$sp,48
243     .cpstore  16
244     sw        $31,40($sp)
245     sw        $fp,36($sp)
246     sw        $28,32($sp)
247     move      $fp,$sp
248     sw        $4,48($fp)
249     lw        $2,48($fp)
250     bne       $2,$0,$L21
251     li        $2,1                # 0x1
252     sw        $2,24($fp)
253     b         $L20
254 $L21:
255     lw        $4,48($fp)
256     la        $5,$LC20
257     la        $25,strcmp
258     jal       $31,$25
259     beq       $2,$0,$L23
260     lw        $4,48($fp)
261     la        $5,$LC21
262     la        $25,strcmp
263     jal       $31,$25
264     beq       $2,$0,$L23
265     lw        $4,48($fp)
266     la        $5,$LC22
267     la        $25,strcmp
268     jal       $31,$25
269     beq       $2,$0,$L23
270     lw        $4,48($fp)
271     la        $5,$LC23
272     la        $25,strcmp
273     jal       $31,$25

```

```

274     bne $2,$0,$L22
275 $L23:
276     li  $2,1          # 0x1
277     sw  $2,24($fp)
278     b   $L20
279 $L22:
280     sw  $0,24($fp)
281 $L20:
282     lw  $2,24($fp)
283     move $sp,$fp
284     lw  $31,40($sp)
285     lw  $fp,36($sp)
286     addu $sp,$sp,48
287     j   $31
288     .end  validateStreamName
289     .size  validateStreamName, .-validateStreamName
290     .rdata
291     .align 2
292 $LC24:
293     .ascii "ERROR: Invalid input stream.\n\000"
294     .align 2
295 $LC25:
296     .ascii "-\000"
297     .align 2
298 $LC26:
299     .ascii "rb\000"
300     .align 2
301 $LC27:
302     .ascii "ERROR: Can't open input stream.\n\000"
303     .text
304     .align 2
305     .globl optInput
306     .ent  optInput
307 optInput:
308     .frame $fp,48,$31      # vars= 8, regs= 4/0, args= 16, ←
        extra= 8
309     .mask 0xd0010000,-4
310     .fmask 0x00000000,0
311     .set  noreorder
312     .cpload $25
313     .set  reorder
314     subu  $sp,$sp,48
315     .cpstore 16
316     sw  $31,44($sp)
317     sw  $fp,40($sp)
318     sw  $28,36($sp)
319     sw  $16,32($sp)

```

```

320     move    $fp,$sp
321     sw     $4,48($fp)
322     sw     $5,52($fp)
323     lw     $4,48($fp)
324     la     $25,validateStreamName
325     jal    $31,$25
326     move    $3,$2
327     li     $2,1                # 0x1
328     bne    $3,$2,$L25
329     la     $4,__$sF+176
330     la     $5,$LC24
331     la     $25,fprintf
332     jal    $31,$25
333     li     $2,1                # 0x1
334     sw     $2,24($fp)
335     b     $L24
336 $L25:
337     lw     $4,48($fp)
338     la     $5,$LC25
339     la     $25,strcmp
340     jal    $31,$25
341     bne    $2,$0,$L26
342     lw     $3,52($fp)
343     la     $2,__$sF
344     sw     $2,4($3)
345     b     $L27
346 $L26:
347     lw     $16,52($fp)
348     lw     $4,48($fp)
349     la     $5,$LC26
350     la     $25,fopen
351     jal    $31,$25
352     sw     $2,4($16)
353 $L27:
354     lw     $2,52($fp)
355     lw     $2,4($2)
356     bne    $2,$0,$L28
357     la     $4,__$sF+176
358     la     $5,$LC27
359     la     $25,fprintf
360     jal    $31,$25
361     li     $2,1                # 0x1
362     sw     $2,24($fp)
363     b     $L24
364 $L28:
365     sw     $0,24($fp)
366 $L24:

```

```

367     lw    $2,24($fp)
368     move   $sp,$fp
369     lw    $31,44($sp)
370     lw    $fp,40($sp)
371     lw    $16,32($sp)
372     addu   $sp,$sp,48
373     j      $31
374     .end    optInput
375     .size   optInput, .-optInput
376     .rdata
377     .align  2
378 $LC28:
379     .ascii  "ERROR: Invalid output stream.\n\000"
380     .align  2
381 $LC29:
382     .ascii  "wb\000"
383     .align  2
384 $LC30:
385     .ascii  "ERROR: Can't open output stream.\n\000"
386     .text
387     .align  2
388     .globl  optOutput
389     .ent    optOutput
390 optOutput:
391     .frame  $fp,48,$31      # vars= 8, regs= 4/0, args= 16, ←
        extra= 8
392     .mask   0xd0010000,-4
393     .fmask  0x00000000,0
394     .set     noreorder
395     .cpload  $25
396     .set     reorder
397     subu     $sp,$sp,48
398     .cpstore 16
399     sw    $31,44($sp)
400     sw    $fp,40($sp)
401     sw    $28,36($sp)
402     sw    $16,32($sp)
403     move   $fp,$sp
404     sw    $4,48($fp)
405     sw    $5,52($fp)
406     lw    $4,48($fp)
407     la    $25,validateStreamName
408     jal   $31,$25
409     move   $3,$2
410     li     $2,1             # 0x1
411     bne    $3,$2,$L30
412     la     $4, __sF+176

```

```

413     la    $5,$LC28
414     la    $25,fprintf
415     jal   $31,$25
416     li    $2,1                # 0x1
417     sw    $2,24($fp)
418     b     $L29
419 $L30:
420     lw    $4,48($fp)
421     la    $5,$LC25
422     la    $25,strcmp
423     jal   $31,$25
424     bne   $2,$0,$L31
425     lw    $3,52($fp)
426     la    $2, __sF+88
427     sw    $2,8($3)
428     b     $L32
429 $L31:
430     lw    $16,52($fp)
431     lw    $4,48($fp)
432     la    $5,$LC29
433     la    $25,fopen
434     jal   $31,$25
435     sw    $2,8($16)
436 $L32:
437     lw    $2,52($fp)
438     lw    $2,8($2)
439     bne   $2,$0,$L33
440     la    $4, __sF+176
441     la    $5,$LC30
442     la    $25,fprintf
443     jal   $31,$25
444     li    $2,1                # 0x1
445     sw    $2,24($fp)
446     b     $L29
447 $L33:
448     sw    $0,24($fp)
449 $L29:
450     lw    $2,24($fp)
451     move   $sp,$fp
452     lw    $31,44($sp)
453     lw    $fp,40($sp)
454     lw    $16,32($sp)
455     addu   $sp,$sp,48
456     j      $31
457     .end   optOutput
458     .size  optOutput, .-optOutput
459     .rdata

```

```

460     .align    2
461 $LC31:
462     .ascii    "encode\000"
463     .align    2
464 $LC32:
465     .ascii    "decode\000"
466     .text
467     .align    2
468     .globl    optAction
469     .ent      optAction
470 optAction:
471     .frame    $fp,48,$31          # vars= 8, regs= 3/0, args= 16, ←
472         extra= 8
473     .mask     0xd0000000,-8
474     .fmask    0x00000000,0
475     .set      noreorder
476     .cpload   $25
477     .set      reorder
478     subu      $sp,$sp,48
479     .cprestore 16
480     sw        $31,40($sp)
481     sw        $fp,36($sp)
482     sw        $28,32($sp)
483     move      $fp,$sp
484     sw        $4,48($fp)
485     sw        $5,52($fp)
486     lw        $2,48($fp)
487     bne       $2,$0,$L35
488     la        $4,__sF+176
489     la        $5,$LC7
490     la        $25,fprintf
491     jal       $31,$25
492     li        $2,1                # 0x1
493     sw        $2,24($fp)
494     b         $L34
495 $L35:
496     lw        $4,48($fp)
497     la        $5,$LC31
498     la        $25,strcmp
499     jal       $31,$25
500     bne       $2,$0,$L36
501     lw        $3,52($fp)
502     la        $2,$LC31
503     sw        $2,0($3)
504     b         $L37
505 $L36:
506     lw        $4,48($fp)

```

```

506     la    $5,$LC32
507     la    $25,strcmp
508     jal   $31,$25
509     bne   $2,$0,$L38
510     lw    $3,52($fp)
511     la    $2,$LC32
512     sw    $2,0($3)
513     b     $L37
514 $L38:
515     la    $4, __sF+176
516     la    $5,$LC7
517     la    $25,fprintf
518     jal   $31,$25
519     li    $2,1           # 0x1
520     sw    $2,24($fp)
521     b     $L34
522 $L37:
523     sw    $0,24($fp)
524 $L34:
525     lw    $2,24($fp)
526     move   $sp,$fp
527     lw    $31,40($sp)
528     lw    $fp,36($sp)
529     addu   $sp,$sp,48
530     j      $31
531     .end    optAction
532     .size   optAction, .-optAction
533     .rdata
534     .align  2
535 $LC33:
536     .ascii  "Vhi:o:a:\000"
537     .text
538     .align  2
539     .globl  parseCmdline
540     .ent    parseCmdline
541 parseCmdline:
542     .frame  $fp,80,$31      # vars= 32, regs= 3/0, args= 24, ←
543         extra= 8
544     .mask   0xd0000000,-8
545     .fmask  0x00000000,0
546     .set    noreorder
547     .cpld   $25
548     .set    reorder
549     subu    $sp,$sp,80
550     .cprestore 24
551     sw     $31,72($sp)
552     sw     $fp,68($sp)

```



```

552     sw    $28,64($sp)
553     move   $fp,$sp
554     sw    $4,80($fp)
555     sw    $5,84($fp)
556     sw    $6,88($fp)
557     sw    $0,32($fp)
558     li     $2,1           # 0x1
559     sw    $2,40($fp)
560     lw    $2,84($fp)
561     lw    $2,0($2)
562     sw    $2,44($fp)
563     lw    $3,88($fp)
564     la     $2,$LC31
565     sw    $2,0($3)
566     lw    $3,88($fp)
567     la     $2, __sF
568     sw    $2,4($3)
569     lw    $3,88($fp)
570     la     $2, __sF+88
571     sw    $2,8($3)
572     la     $2,$LC33
573     sw    $2,48($fp)
574 $L41:
575     addu   $2,$fp,32
576     sw    $2,16($sp)
577     lw    $4,80($fp)
578     lw    $5,84($fp)
579     lw    $6,48($fp)
580     la     $7,cmdOptions
581     la     $25,getopt_long
582     jal    $31,$25
583     sw    $2,36($fp)
584     lw    $3,36($fp)
585     li     $2,-1          # 0xffffffffffffffff
586     bne    $3,$2,$L43
587     b      $L42
588 $L43:
589     lw    $2,36($fp)
590     addu   $2,$2,-63
591     sw    $2,56($fp)
592     lw    $3,56($fp)
593     sltu   $2,$3,49
594     beq    $2,$0,$L44
595     lw    $2,56($fp)
596     sll    $3,$2,2
597     la     $2,$L51
598     addu   $2,$3,$2

```

```
599     lw    $2,0($2)
600     .cpadd $2
601     j     $2
602     .rdata
603     .align 2
604 $L51:
605     .gpword $L50
606     .gpword $L44
607     .gpword $L44
608     .gpword $L44
609     .gpword $L44
610     .gpword $L44
611     .gpword $L44
612     .gpword $L44
613     .gpword $L44
614     .gpword $L44
615     .gpword $L44
616     .gpword $L44
617     .gpword $L44
618     .gpword $L44
619     .gpword $L44
620     .gpword $L44
621     .gpword $L44
622     .gpword $L44
623     .gpword $L44
624     .gpword $L44
625     .gpword $L44
626     .gpword $L44
627     .gpword $L44
628     .gpword $L45
629     .gpword $L44
630     .gpword $L44
631     .gpword $L44
632     .gpword $L44
633     .gpword $L44
634     .gpword $L44
635     .gpword $L44
636     .gpword $L44
637     .gpword $L44
638     .gpword $L44
639     .gpword $L49
640     .gpword $L44
641     .gpword $L44
642     .gpword $L44
643     .gpword $L44
644     .gpword $L44
645     .gpword $L44
```

```
646      .gpword $L46
647      .gpword $L47
648      .gpword $L44
649      .gpword $L44
650      .gpword $L44
651      .gpword $L44
652      .gpword $L44
653      .gpword $L48
654      .text
655 $L45:
656      la    $25,optVersion
657      jal   $31,$25
658      b     $L44
659 $L46:
660      lw    $4,44($fp)
661      la    $25,optHelp
662      jal   $31,$25
663      b     $L44
664 $L47:
665      lw    $4,optarg
666      lw    $5,88($fp)
667      la    $25,optInput
668      jal   $31,$25
669      sw    $2,40($fp)
670      b     $L44
671 $L48:
672      lw    $4,optarg
673      lw    $5,88($fp)
674      la    $25,optOutput
675      jal   $31,$25
676      sw    $2,40($fp)
677      b     $L44
678 $L49:
679      lw    $4,optarg
680      lw    $5,88($fp)
681      la    $25,optAction
682      jal   $31,$25
683      sw    $2,40($fp)
684      b     $L44
685 $L50:
686      li    $2,1                # 0x1
687      sw    $2,40($fp)
688 $L44:
689      lw    $3,40($fp)
690      li    $2,1                # 0x1
691      bne   $3,$2,$L41
692      li    $2,1                # 0x1
```

```
693     sw    $2,52($fp)
694     b     $L40
695 $L42:
696     sw    $0,52($fp)
697 $L40:
698     lw    $2,52($fp)
699     move   $sp,$fp
700     lw    $31,72($sp)
701     lw    $fp,68($sp)
702     addu   $sp,$sp,80
703     j     $31
704     .end   parseCmdline
705     .size  parseCmdline, .-parseCmdline
706     .ident "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

C.0.12. encoders

```

1      .file      1  "encoder.c"
2      .section   .mdebug.abi32
3      .previous
4      .abicalls
5      .rdata
6      .align     2
7      .type      translationTableB64, @object
8      .size      translationTableB64, 64
9 translationTableB64:
10     .ascii      "↵
                ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123↵
                "
11     .ascii      "456789+/"
12     .local      tailByte.0
13     .comm       tailByte.0,1,1
14     .data
15     .type      bitMask.1, @object
16     .size      bitMask.1, 1
17 bitMask.1:
18     .byte      -4
19     .align     2
20     .type      shiftRightBit.2, @object
21     .size      shiftRightBit.2, 4
22 shiftRightBit.2:
23     .word      2
24     .rdata
25     .align     2
26 $LC0:
27     .ascii      "=\000"
28     .text
29     .align     2
30     .globl      base256ToBase64
31     .ent        base256ToBase64
32 base256ToBase64:
33     .frame      $fp,48,$31      # vars= 8, regs= 3/0, args= 16, ↵
                extra= 8
34     .mask       0xd0000000,-8
35     .fmask      0x00000000,0
36     .set        noreorder
37     .cpload     $25
38     .set        reorder
39     subu        $sp,$sp,48
40     .cpstore    16
41     sw          $31,40($sp)

```

```

42     sw    $fp,36($sp)
43     sw    $28,32($sp)
44     move   $fp,$sp
45     sw    $4,48($fp)
46     move   $2,$5
47     sb    $2,24($fp)
48     sb    $0,25($fp)
49     sb    $0,26($fp)
50     sb    $0,27($fp)
51     lbu   $2,tailByte.0
52     sb    $2,26($fp)
53     lb    $3,24($fp)
54     li    $2,-1           # 0xffffffffffffffff
55     bne   $3,$2,$L6
56     lw    $3,shiftRightBit.2
57     li    $2,6           # 0x6
58     bne   $3,$2,$L7
59     lbu   $2,26($fp)
60     sb    $2,25($fp)
61     lbu   $3,25($fp)
62     la    $2,translationTableB64
63     addu   $2,$3,$2
64     lw    $4,48($fp)
65     move   $5,$2
66     li    $6,1           # 0x1
67     la    $25,strncpy
68     jal   $31,$25
69     lw    $4,48($fp)
70     la    $5,$LC0
71     li    $6,1           # 0x1
72     la    $25,strncat
73     jal   $31,$25
74     lbu   $2,27($fp)
75     addu   $2,$2,2
76     andi   $2,$2,0x00ff
77     sw    $2,28($fp)
78     b     $L5
79 $L7:
80     lw    $3,shiftRightBit.2
81     li    $2,4           # 0x4
82     bne   $3,$2,$L9
83     lbu   $2,26($fp)
84     sb    $2,25($fp)
85     lbu   $3,25($fp)
86     la    $2,translationTableB64
87     addu   $2,$3,$2
88     lw    $4,48($fp)

```

```

89     move    $5,$2
90     li      $6,1                # 0x1
91     la      $25, strncpy
92     jal     $31,$25
93     lw      $4,48($fp)
94     la      $5,$LC0
95     li      $6,1                # 0x1
96     la      $25, strncat
97     jal     $31,$25
98     lw      $4,48($fp)
99     la      $5,$LC0
100    li      $6,1                # 0x1
101    la      $25, strncat
102    jal     $31,$25
103    lbu     $2,27($fp)
104    addu     $2,$2,3
105    andi     $2,$2,0x00ff
106    sw      $2,28($fp)
107    b       $L5
108 $L9:
109    lbu     $2,27($fp)
110    sw      $2,28($fp)
111    b       $L5
112 $L6:
113    lbu     $3,24($fp)
114    lbu     $2, bitMask.1
115    and     $2,$3,$2
116    sb      $2,25($fp)
117    lbu     $3,25($fp)
118    lw      $2, shiftRightBit.2
119    sra     $2,$3,$2
120    sb      $2,25($fp)
121    lbu     $2, bitMask.1
122    nor     $2,$0,$2
123    lbu     $3,24($fp)
124    and     $2,$3,$2
125    sb      $2, tailByte.0
126    lbu     $4, tailByte.0
127    li      $3,6                # 0x6
128    lw      $2, shiftRightBit.2
129    subu     $2,$3,$2
130    sll     $2,$4,$2
131    sb      $2, tailByte.0
132    lbu     $3,25($fp)
133    lbu     $2,26($fp)
134    or      $2,$3,$2
135    sb      $2,25($fp)

```

```

136     lbu $3,25($fp)
137     la  $2,translationTableB64
138     addu $2,$3,$2
139     lw  $4,48($fp)
140     move $5,$2
141     li  $6,1          # 0x1
142     la  $25,strncpy
143     jal $31,$25
144     lbu $2,27($fp)
145     addu $2,$2,1
146     sb  $2,27($fp)
147     lw  $2,shiftRightBit.2
148     addu $2,$2,2
149     sw  $2,shiftRightBit.2
150     lbu $2,bitMask.1
151     sll $2,$2,2
152     sb  $2,bitMask.1
153     andi $2,$2,0x00ff
154     bne $2,$0,$L11
155     li  $2,-4          # 0xfffffffffffffffffc
156     sb  $2,bitMask.1
157     li  $2,2           # 0x2
158     sw  $2,shiftRightBit.2
159     lbu $3,tailByte.0
160     la  $2,translationTableB64
161     addu $2,$3,$2
162     lw  $4,48($fp)
163     move $5,$2
164     li  $6,1          # 0x1
165     la  $25,strncat
166     jal $31,$25
167     lbu $2,27($fp)
168     addu $2,$2,1
169     sb  $2,27($fp)
170     sb  $0,tailByte.0
171 $L11:
172     lbu $2,27($fp)
173     sw  $2,28($fp)
174 $L5:
175     lw  $2,28($fp)
176     move $sp,$fp
177     lw  $31,40($sp)
178     lw  $fp,36($sp)
179     addu $sp,$sp,48
180     j   $31
181     .end    base256ToBase64
182     .size   base256ToBase64, .-base256ToBase64

```



```

183     .rdata
184     .align 2
185 $LC1:
186     .ascii "Input error when reading stream.\n\000"
187     .align 2
188 $LC2:
189     .ascii "\n\000"
190     .align 2
191 $LC3:
192     .ascii "Output error when writing stream.\n\000"
193     .text
194     .align 2
195     .globl encode
196     .ent encode
197 encode:
198     .frame $fp,72,$31      # vars= 32, regs= 3/0, args= 16, ←
199     extra= 8
200     .mask 0xd0000000,-8
201     .fmask 0x00000000,0
202     .set noreorder
203     .cpload $25
204     .set reorder
205     subu $sp,$sp,72
206     .cpstore 16
207     sw $31,64($sp)
208     sw $fp,60($sp)
209     sw $28,56($sp)
210     move $fp,$sp
211     sw $4,72($fp)
212     sw $0,32($fp)
213     sb $0,36($fp)
214     sb $0,40($fp)
215     sb $0,41($fp)
216 $L13:
217     addu $2,$fp,32
218     move $4,$2
219     move $5,$0
220     li $6,5          # 0x5
221     la $25,memset
222     jal $31,$25
223     lw $2,72($fp)
224     lw $3,4($2)
225     lw $2,72($fp)
226     lw $2,4($2)
227     lw $2,4($2)
228     addu $2,$2,-1
229     sw $2,4($3)

```

```

229     bgez     $2,$L16
230     lw      $2,72($fp)
231     lw      $4,4($2)
232     la      $25, __srget
233     jal     $31,$25
234     sb      $2,48($fp)
235     b       $L17
236 $L16:
237     lw      $2,72($fp)
238     lw      $4,4($2)
239     lw      $2,0($4)
240     move     $3,$2
241     lbu     $3,0($3)
242     sb      $3,48($fp)
243     addu     $2,$2,1
244     sw      $2,0($4)
245 $L17:
246     lbu     $2,48($fp)
247     sb      $2,24($fp)
248     lw      $2,72($fp)
249     lw      $2,4($2)
250     lhu     $2,12($2)
251     srl     $2,$2,6
252     andi     $2,$2,0x1
253     beq     $2,$0,$L18
254     la      $4, __sF+176
255     la      $5,$LC1
256     la      $25,fprintf
257     jal     $31,$25
258     li      $2,1                # 0x1
259     sw      $2,44($fp)
260     b       $L12
261 $L18:
262     addu     $2,$fp,32
263     lb      $3,24($fp)
264     move     $4,$2
265     move     $5,$3
266     la      $25,base256ToBase64
267     jal     $31,$25
268     sb      $2,41($fp)
269     lbu     $3,40($fp)
270     lbu     $2,41($fp)
271     addu     $2,$3,$2
272     slt     $2,$2,77
273     beq     $2,$0,$L19
274     lbu     $2,40($fp)
275     lbu     $3,41($fp)

```

```
276     addu    $2,$2,$3
277     sb      $2,40($fp)
278     b       $L20
279 $L19:
280     lw      $2,72($fp)
281     la      $4,$LC2
282     lw      $5,8($2)
283     la      $25,fputs
284     jal     $31,$25
285     lw      $2,72($fp)
286     lw      $2,8($2)
287     lhu     $2,12($2)
288     srl     $2,$2,6
289     andi     $2,$2,0x1
290     beq     $2,$0,$L21
291     la      $4, __sF+176
292     la      $5,$LC3
293     la      $25,fprintf
294     jal     $31,$25
295     li      $2,1                # 0x1
296     sw      $2,44($fp)
297     b       $L12
298 $L21:
299     lbu     $2,41($fp)
300     sb      $2,40($fp)
301 $L20:
302     addu    $2,$fp,32
303     lw      $3,72($fp)
304     move     $4,$2
305     lw      $5,8($3)
306     la      $25,fputs
307     jal     $31,$25
308     lw      $2,72($fp)
309     lw      $2,8($2)
310     lhu     $2,12($2)
311     srl     $2,$2,6
312     andi     $2,$2,0x1
313     beq     $2,$0,$L15
314     la      $4, __sF+176
315     la      $5,$LC3
316     la      $25,fprintf
317     jal     $31,$25
318     li      $2,1                # 0x1
319     sw      $2,44($fp)
320     b       $L12
321 $L15:
322     lb      $3,24($fp)
```

```
323     li    $2,-1           # 0xffffffffffffffff
324     bne   $3,$2,$L13
325     sw    $0,44($fp)
326 $L12:
327     lw    $2,44($fp)
328     move   $sp,$fp
329     lw    $31,64($sp)
330     lw    $fp,60($sp)
331     addu   $sp,$sp,72
332     j     $31
333     .end   encode
334     .size  encode,.-encode
335     .ident "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

C.0.13. decoders

```

1  .file    1  "decoder.c"
2  .section .mdebug.abi32
3  .previous
4  .abicalls
5  .rdata
6  .align   2
7  .type    translationTableB64, @object
8  .size    translationTableB64, 64
9  translationTableB64:
10  .ascii   "↵
           ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123↵
           "
11  .ascii   "456789+/"
12  .align   2
13 $LC0:
14  .ascii   "ERROR: Character is not in Base64 Table\n\000"
15  .text
16  .align   2
17  .globl   base64ToBase256
18  .ent     base64ToBase256
19 base64ToBase256:
20  .frame   $fp,80,$31      # vars= 40, regs= 3/0, args= 16, ↵
           extra= 8
21  .mask    0xd0000000,-8
22  .fmask    0x00000000,0
23  .set     noreorder
24  .cpload  $25
25  .set     reorder
26  subu     $sp,$sp,80
27  .cpstore 16
28  sw       $31,72($sp)
29  sw       $fp,68($sp)
30  sw       $28,64($sp)
31  move     $fp,$sp
32  sw       $4,80($fp)
33  sw       $5,84($fp)
34  li       $2,-16777216      # 0xfffffffffff000000
35  sw       $2,24($fp)
36  sw       $0,28($fp)
37  sw       $0,32($fp)
38  sw       $0,40($fp)
39  sw       $0,48($fp)
40  sw       $0,52($fp)
41  sb       $0,56($fp)

```

```
42     sw    $0,28($fp)
43 $L6:
44     lw    $2,28($fp)
45     sltu   $2,$2,4
46     bne   $2,$0,$L9
47     b     $L7
48 $L9:
49     lw    $3,84($fp)
50     lw    $2,28($fp)
51     addu   $2,$3,$2
52     lbu   $3,0($2)
53     li    $2,61           # 0x3d
54     bne   $3,$2,$L10
55     addu   $2,$fp,40
56     lw    $3,28($fp)
57     addu   $2,$2,$3
58     sb    $0,0($2)
59     b     $L8
60 $L10:
61     sw    $0,32($fp)
62 $L11:
63     lw    $2,32($fp)
64     sltu   $2,$2,64
65     bne   $2,$0,$L14
66     b     $L12
67 $L14:
68     lw    $3,84($fp)
69     lw    $2,28($fp)
70     addu   $2,$3,$2
71     lbu   $4,0($2)
72     lw    $3,32($fp)
73     la    $2,translationTableB64
74     addu   $2,$3,$2
75     lb    $2,0($2)
76     bne   $4,$2,$L13
77     addu   $3,$fp,40
78     lw    $2,28($fp)
79     addu   $3,$3,$2
80     lbu   $2,32($fp)
81     sb    $2,0($3)
82     b     $L12
83 $L13:
84     lw    $2,32($fp)
85     addu   $2,$2,1
86     sw    $2,32($fp)
87     b     $L11
88 $L12:
```

```

89     lw    $2,32($fp)
90     sltu   $2,$2,64
91     bne    $2,$0,$L8
92     la     $4,__$sF+176
93     la     $5,$LC0
94     la     $25,fprintf
95     jal    $31,$25
96     li     $2,1                # 0x1
97     sw     $2,60($fp)
98     b      $L5
99 $L8:
100    lw     $2,28($fp)
101    addu    $2,$2,1
102    sw     $2,28($fp)
103    b      $L6
104 $L7:
105    sw     $0,28($fp)
106 $L17:
107    lw     $2,28($fp)
108    sltu   $2,$2,4
109    bne    $2,$0,$L20
110    b      $L18
111 $L20:
112    lbu    $2,56($fp)
113    addu    $2,$2,2
114    sb     $2,56($fp)
115    addu    $3,$fp,40
116    lw     $2,28($fp)
117    addu    $2,$3,$2
118    lbu    $2,0($2)
119    sw     $2,48($fp)
120    li     $3,3                # 0x3
121    lw     $2,28($fp)
122    subu    $2,$3,$2
123    sll    $3,$2,3
124    lw     $2,48($fp)
125    sll    $2,$2,$3
126    sw     $2,48($fp)
127    lbu    $3,56($fp)
128    lw     $2,48($fp)
129    sll    $2,$2,$3
130    sw     $2,48($fp)
131    lw     $3,52($fp)
132    lw     $2,48($fp)
133    or     $2,$3,$2
134    sw     $2,52($fp)
135    lw     $2,28($fp)

```

```

136     addu    $2,$2,1
137     sw     $2,28($fp)
138     b      $L17
139 $L18:
140     sw     $0,28($fp)
141 $L21:
142     lw     $3,52($fp)
143     lw     $2,24($fp)
144     and    $2,$3,$2
145     sw     $2,48($fp)
146     li     $3,3                # 0x3
147     lw     $2,28($fp)
148     subu   $2,$3,$2
149     sll    $3,$2,3
150     lw     $2,48($fp)
151     srl    $2,$2,$3
152     sw     $2,48($fp)
153     lw     $3,80($fp)
154     lw     $2,28($fp)
155     addu   $3,$3,$2
156     lbu    $2,48($fp)
157     sb     $2,0($3)
158     lw     $2,24($fp)
159     srl    $2,$2,8
160     sw     $2,24($fp)
161     lw     $2,28($fp)
162     addu   $2,$2,1
163     sw     $2,28($fp)
164     lw     $2,48($fp)
165     beq    $2,$0,$L22
166     lw     $2,28($fp)
167     sltu   $2,$2,4
168     bne    $2,$0,$L21
169 $L22:
170     sw     $0,60($fp)
171 $L5:
172     lw     $2,60($fp)
173     move   $sp,$fp
174     lw     $31,72($sp)
175     lw     $fp,68($sp)
176     addu   $sp,$sp,80
177     j      $31
178     .end   base64ToBase256
179     .size  base64ToBase256, .-base64ToBase256
180     .rdata
181     .align 2
182 $LC1:

```



```

183     .ascii  "Input error when reading stream.\n\000"
184     .align  2
185 $LC2:
186     .ascii  "Output error when writing stream.\n\000"
187     .text
188     .align  2
189     .globl  decode
190     .ent    decode
191 decode:
192     .frame  $fp,80,$31          # vars= 40, regs= 3/0, args= 16, ←
        extra= 8
193     .mask   0xd0000000,-8
194     .fmask  0x00000000,0
195     .set    noreorder
196     .cpload $25
197     .set    reorder
198     subu    $sp,$sp,80
199     .cpstore 16
200     sw      $31,72($sp)
201     sw      $fp,68($sp)
202     sw      $28,64($sp)
203     move    $fp,$sp
204     sw      $4,80($fp)
205     sw      $0,32($fp)
206     sh      $0,40($fp)
207     sb      $0,42($fp)
208     sb      $0,48($fp)
209 $L27:
210     sb      $0,48($fp)
211 $L30:
212     lbu     $2,48($fp)
213     sltu    $2,$2,4
214     bne     $2,$0,$L33
215     b       $L31
216 $L33:
217     lw      $2,80($fp)
218     lw      $3,4($2)
219     lw      $2,80($fp)
220     lw      $2,4($2)
221     lw      $2,4($2)
222     addu    $2,$2,-1
223     sw      $2,4($3)
224     bgez    $2,$L34
225     lw      $2,80($fp)
226     lw      $4,4($2)
227     la      $25,__srget
228     jal     $31,$25

```

```

229     sb    $2,56($fp)
230     b     $L35
231 $L34:
232     lw    $2,80($fp)
233     lw    $4,4($2)
234     lw    $2,0($4)
235     move   $3,$2
236     lbu   $3,0($3)
237     sb    $3,56($fp)
238     addu   $2,$2,1
239     sw    $2,0($4)
240 $L35:
241     lbu   $2,56($fp)
242     sb    $2,24($fp)
243     lw    $2,80($fp)
244     lw    $2,4($2)
245     lhu   $2,12($2)
246     srl   $2,$2,6
247     andi   $2,$2,0x1
248     beq   $2,$0,$L36
249     la    $4,__$sF+176
250     la    $5,$LC1
251     la    $25,fprintf
252     jal   $31,$25
253     li    $2,1                # 0x1
254     sw    $2,52($fp)
255     b     $L26
256 $L36:
257     lbu   $3,24($fp)
258     li    $2,10                # 0xa
259     beq   $3,$2,$L38
260     lbu   $3,24($fp)
261     li    $2,9                 # 0x9
262     beq   $3,$2,$L38
263     lbu   $3,24($fp)
264     li    $2,32                # 0x20
265     beq   $3,$2,$L38
266     b     $L37
267 $L38:
268     lbu   $2,48($fp)
269     addu   $2,$2,-1
270     sb    $2,48($fp)
271     b     $L32
272 $L37:
273     lbu   $3,48($fp)
274     addu   $2,$fp,32
275     addu   $3,$2,$3

```

```

276     lbu $2,24($fp)
277     sb  $2,0($3)
278     lw  $2,80($fp)
279     lw  $2,4($2)
280     lhu $2,12($2)
281     srl $2,$2,5
282     andi $2,$2,0x1
283     beq $2,$0,$L32
284     lbu $2,48($fp)
285     beq $2,$0,$L41
286     addu $2,$fp,40
287     addu $3,$fp,32
288     move $4,$2
289     move $5,$3
290     la  $25,base64ToBase256
291     jal $31,$25
292     move $3,$2
293     li  $2,1          # 0x1
294     bne $3,$2,$L42
295     li  $2,1          # 0x1
296     sw  $2,52($fp)
297     b   $L26
298 $L42:
299     addu $2,$fp,40
300     lw  $3,80($fp)
301     move $4,$2
302     lw  $5,8($3)
303     la  $25,fputs
304     jal $31,$25
305     lw  $2,80($fp)
306     lw  $2,8($2)
307     lhu $2,12($2)
308     srl $2,$2,6
309     andi $2,$2,0x1
310     beq $2,$0,$L41
311     la  $4, __sF+176
312     la  $5,$LC2
313     la  $25,fprintf
314     jal $31,$25
315     li  $2,1          # 0x1
316     sw  $2,52($fp)
317     b   $L26
318 $L41:
319     sw  $0,52($fp)
320     b   $L26
321 $L32:
322     lbu $2,48($fp)

```

```
323     addu    $2,$2,1
324     sb     $2,48($fp)
325     b      $L30
326 $L31:
327     addu    $2,$fp,40
328     addu    $3,$fp,32
329     move    $4,$2
330     move    $5,$3
331     la     $25,base64ToBase256
332     jal     $31,$25
333     move    $3,$2
334     li     $2,1                # 0x1
335     bne     $3,$2,$L44
336     li     $2,1                # 0x1
337     sw     $2,52($fp)
338     b      $L26
339 $L44:
340     addu    $2,$fp,40
341     lw     $3,80($fp)
342     move    $4,$2
343     lw     $5,8($3)
344     la     $25,fputs
345     jal     $31,$25
346     lw     $2,80($fp)
347     lw     $2,8($2)
348     lhu    $2,12($2)
349     srl    $2,$2,6
350     andi    $2,$2,0x1
351     beq     $2,$0,$L27
352     la     $4, __sF+176
353     la     $5,$LC2
354     la     $25,fprintf
355     jal     $31,$25
356     li     $2,1                # 0x1
357     sw     $2,52($fp)
358 $L26:
359     lw     $2,52($fp)
360     move    $sp,$fp
361     lw     $31,72($sp)
362     lw     $fp,68($sp)
363     addu    $sp,$sp,80
364     j      $31
365     .end    decode
366     .size   decode,.-decode
367     .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```