
Facultad de Ingeniería, Universidad de Buenos Aires
66.20 Organización de Computadoras

Trabajo Práctico 0

Infraestructura básica

Impaglione, Rocio - Padrón: 94178

rocio.impaglione@gmail.com

Monsech, Santiago - Padrón: 92968

santiagomonsech@gmail.com

Trucco, Guido - Padrón: 97980

gftrucco@gmail.com

2do Cuatrimestre de 2018

Índice

1. Introducción	2
2. Desarrollo del Trabajo Práctico	3
2.1. Diseño e Implementación	3
2.2. Compilación del programa	3
2.3. Funcionamiento del programa	3
2.4. Corridas de Prueba	5
3. Código Fuente	9
3.1. Código C	9
3.2. Código MIPS32	17
4. Conclusiones	52
5. Referencias	53
6. Anexo	54
6.1. Enunciado provisto por los profesores del curso	54

1. Introducción

El objetivo del presente trabajo práctico es lograr que los alumnos se familiaricen con las herramientas de software que se utilizarán en la materia. A tal fin, los mismos deberán implementar un programa que resuelva el problema presentado por los profesores en el enunciado del presente trabajo, junto con la documentación correspondiente.

2. Desarrollo del Trabajo Práctico

2.1. Diseño e Implementación

El trabajo práctico pide a los alumnos que desarrollen una aplicación que reciba un texto (que puede ingresarse por medio de un archivo o utilizando la entrada standard) y permita codificar o decodificar el contenido del mismo al protocolo de base64, según lo desee el usuario. Una vez realizado este procesamiento, el programa deberá devolver el resultado por salida standard o a través de un archivo de texto especificado en los parámetros de entrada.

A tales fines, se utilizó el lenguaje C para programar lo pedido. Se buscó respetar las especificaciones dadas por el enunciado entregado por los profesores en lo que respecta a parámetros de entrada, salida y forma de procesamiento.

2.2. Compilación del programa

El archivo fuente del programa se denomina `tp0.c` y se incluye un archivo `decode.c/decode.h` y `encode.c/encode.h` en los cuales se encuentra la logica de codificacion y decodificacion.

Para compilar el programa en sistemas Linux, se debe utilizar el comando

make

en el directorio donde se encuentre dicho archivo, pues incluimos un Makefile para facilitar este proceso.

Para ejecutar en NetBSD, se compila con el comando

gcc -Wall -O0 tp0.c decode.c encode.c -lm -o {archivo ejecutable}

Si se desea obtener el código assembly, hay que ejecutar los comandos

gcc -Wall -O0 -S -mrnames tp0.c -o {archivo salida}
gcc -Wall -O0 -S -mrnames encode.c -o {archivo salida}
gcc -Wall -O0 -S -mrnames decode.c -o {archivo salida}

Una vez realizado el paso anterior, para ejecutar el programa solo basta con situarse en la carpeta donde se haya hecho la compilacion e ingresar

./tp0

con los parámetros que se desee utilizar (los mismos se encuentran especificados en la siguiente seccion).

2.3. Funcionamiento del programa

Como ya se menciona, una vez que se realiza la compilacion detallada en el inciso anterior, el programa esta listo para usarse.

Para correr la aplicacion con los parametros por defecto, solamente basta abrir una terminal y escribir:

./tp0

A continuacion, se le retornara el control al usuario para que pueda ingresar un valor a codificar por teclado (entrada standard). Presionando la tecla Enter o Ctrl + D (end of transmission), indicara a la aplicacion que termino de ingresar texto y lo puede procesar y devolver el resultado. La ejecucion emitira por salida standard el resultado de la decodificacion del stream de texto ingresado. Recordemos que los valores por defecto descriptos en el enunciado son:

- Input: stdin
- Output: stdout
- Action: encode

Si el usuario deseara cambiar el nombre del archivo de entrada por defecto, podra hacerlo a traves del parametro `-i` o `-input`. Por ejemplo:

```
./tp0 -i entrada.txt
```

Un procedimiento similar debe realizarse si se desea reemplazar la salida stdout por una salida a un archivo especifico. El usuario debera utilizar el parametro `-o` o `-output`, seguido del nombre del archivo de salida deseado. En caso de no existir, el mismo sera creado.

Algo importante a notar, es que la aplicacion valida el ingreso de una ruta a un directorio valido. Si el usuario ingresa mal el valor de entrada o salida, el programa finalizara devolviendo error.

```
./tp0 -o directorio/inexistente/archivo.txt
```

```
File path is invalid. Type 'tp0 -h' for help. Program terminated
```

Ademas, la aplicacion admite el ingreso de un parametro `-a` o `-action`, que permite al usuario especificar si se desea hacer una codificacion (encode) o una decodificacion (decode) del stream de texto de entrada.

Por ejemplo,

```
echo -n Man | ./tp0 -a encode
```

De esta forma se obtendra el string codificado a base64, y el mismo sera retornado por salida estandar. El resultado sera:

```
TWФу
```

Si quisieramos decodificar un mensaje codificado:

```
echo -n aGVsbG8gd29ybGQK | ./tp0 -a decode
```

El resultado de la decodificacion sera

```
hello world
```

En caso de que existan dudas sobre el uso de la aplicacion, el usuario puede escribir

```
./tp0 -h
```

y de este modo podra acceder a la opcion de ayuda del programa

TP0 Organizacion de computadoras - HELP

Usage:

tp0 -h Displays help and usage of the application

tp0 -V Displays version of the application

tp0 Options:

-i -input Set the location of the input file

-o -output Set the location of the output file

-a -action Program action: encode (default) or decode

Examples:

tp0 -a encode -i /input -o /output

tp0 -a decode

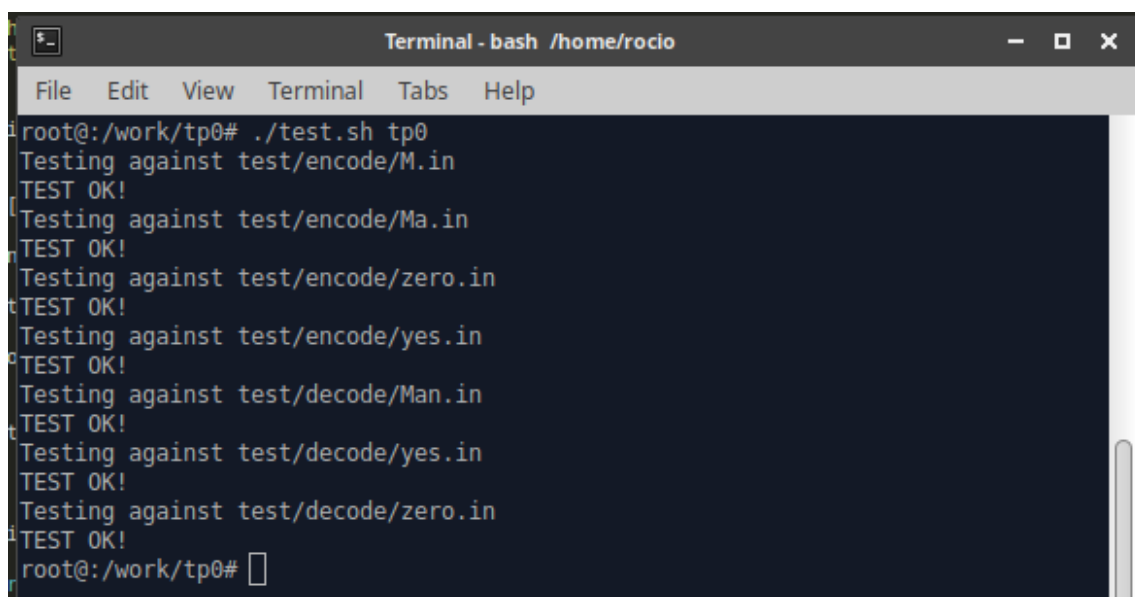
La version de la aplicacion puede verse utilizando el parametro -V:

tp0 -V TP0 Organizacion de computadoras - VERSION: 1.0

2.4. Corridas de Prueba

En este apartado se presentan las corridas de prueba que muestran el funcionamiento de la aplicacion desarrollada.

Incluimos junto con la resolucion del trabajo practico un archivo bash (test.sh) que realiza los tests que se muestran a continuacion de forma automatica.

A screenshot of a terminal window titled "Terminal - bash /home/rocio". The terminal shows the execution of a script named test.sh. The user is at the prompt root@:/work/tp0#. The script runs several tests, each starting with "Testing against test/encode/M.in" or "test/decode/Man.in", followed by "TEST OK!". The tests include encode/M.in, encode/Ma.in, encode/zero.in, encode/yes.in, decode/Man.in, decode/yes.in, and decode/zero.in. The terminal ends with the prompt root@:/work/tp0# and a cursor.

- Codificacion de archivo vacio:

```
Terminal - bash /home/rocio
File Edit View Terminal Tabs Help
root@:/work/tp0# touch /tmp/zero.txt
root@:/work/tp0# ./tp0 -a encode -i /tmp/zero.txt -o /tmp/zero.txt.b64
root@:/work/tp0# ls -l /tmp/zero.txt.b64
-rw-r--r-- 1 root wheel 0 Sep 26 23:22 /tmp/zero.txt.b64
root@:/work/tp0#
```

- Codificación y decodificación básica:

```
Terminal - bash /home/rocio
File Edit View Terminal Tabs Help
root@:/work/tp0# echo -n M | ./tp0
TQ==root@:/work/tp0# echo -n Ma | ./tp0
TWE=root@:/work/tp0# echo -n Man | ./tp0
TWFuroot@:/work/tp0# echo Man | tp0 | tp0 -a decode
Man
root@:/work/tp0#
```

- Verificación bit a bit:

```
Terminal - bash /home/rocio
File Edit View Terminal Tabs Help
root@:/work/tp0# echo xyz | ./tp0 | ./tp0 -a decode | od -t c
0000000  x  y  z  \n
0000004
root@:/work/tp0#
```

- Codificamos 1024 bytes y chequeamos la cantidad:

3. Código Fuente

3.1. Código C

constants.h

```
#ifndef TP0_CONSTANTS_H
#define TP0_CONSTANTS_H

#define ERR_EMPTY -1
#define ERR_INVALID_CHARS -2
#define ERR_INVALID_FORMAT -3
#define ERR_INVALID_FILE_PATH -4
#define ERR_INVALID_PARAMETER -5
#define ERR_INVALID_ACTION_TYPE -6

#define BLOCK_SIZE_INPUT_ENCODING 3
#define BLOCK_SIZE_INPUT_DECODING 4
#define BLOCK_SIZE_OUTPUT_ENCODING 5
#define INVALID_CHAR -1
#define WHITESPACE -2
#define EQUALS -3

extern const unsigned char encodingTable [];
extern const char decodingTable [];

void showVersion(void);
void showHelp();
void showError(int);
void validate(int);
int checkForInvalidCharacters(char*);
int parseAction(char*, char**);
int checkForPath(char*, FILE*, char**, char*);
int readInput(FILE*, unsigned char*, int, int);
void write_partial(unsigned char* processedOutput, FILE* output,
                  char* path);

#endif //TP0_CONSTANTS_H
```

tp0.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <unistd.h>
#include <getopt.h>
#include <memory.h>
#include "encode.h"
#include "decode.h"
#include "constants.h"

typedef struct receivedParameters {
    FILE* output;
    FILE* input;
    char* path_to_output;
    char* path_to_input;
    char* action;
} parameters_t;
```

[illegible]

```

int parseAction(char* action, char **actionToSave){
    if (action == NULL) return ERR_EMPTY;
    if (strcmp(action, "decode") == 0 && (strcmp(action, "encode") == 0))
        return ERR_INVALID_ACTION_TYPE;

    *actionToSave = action;

    return 0;
};

parameters_t getParams(int argc, char **argv){

    int ch;
    parameters_t receivedParameters;

    //defaults
    receivedParameters.output = stdout;
    receivedParameters.path_to_output = NULL;
    receivedParameters.input = stdin;
    receivedParameters.path_to_input = NULL;
    receivedParameters.action = "encode";

    // loop over all of the options
    while ((ch = getopt_long(argc, argv, "hVo:i:a:", long_options, NULL))
        != -1) {
        switch (ch){
            case 'o':
                validate(checkForPath(optarg, receivedParameters.output, &
                    receivedParameters.path_to_output, "out"));
                break;
            case 'i':
                validate(checkForPath(optarg, receivedParameters.input, &
                    receivedParameters.path_to_input, "in"));
                break;
            case 'a':
                validate(parseAction(optarg, &receivedParameters.action));
                break;
            case 'V':
                showVersion();
                exit(0);
                break;
            case 'h':
                showHelp();
                exit(0);
                break;
            case '?':
                if (optopt == 'o' || optopt == 'i' || optopt == 'a') {
                    fprintf(stderr, "No arguments provided for option -%c
                        .\n", optopt);
                } else if (isprint(optopt)) {
                    fprintf(stderr, "Unknown option ‘-%c’.\n", optopt);
                } else {
                    fprintf(stderr, "Unknown option ‘\\x%x’.\n", optopt);
                }
            default:
                showHelp();
                exit(1);
        }
    }
}

```

```

    }

    if (!(stdin)) {
        showHelp();
        exit(1);
    }

    return receivedParameters;
}

void validate(int result){
    if (result < 0){
        showError(result);
        exit(1);
    }
}

void showHelp(){
    printf("%s\n", "TP0 Organizacion de computadoras - HELP");
    printf("%s\n", "Usage: ");
    printf("%s\n", "tp0 -h    Displays help and usage of the application");
    printf("%s\n", "tp0 -V    Displays version of the application");
    printf("%s\n", "tp0 Options:");
    printf("%s\n", "-i --input    Set the location of the input file");
    printf("%s\n", "-o --output    Set the location of the output file");
    printf("%s\n", "-a --action    Program action: encode (default) or
        decode");
    printf("%s\n", "Examples: ");
    printf("%s\n", "    tp0 -a encode -i ~/input -o ~/output");
    printf("%s\n", "    tp0 -a decode");
}

void showVersion(){
    printf("%s\n", "TP0 Organizacion de computadoras - VERSION: 1.0");
}

void showError(int errorCode) {
    if (errorCode == ERR_EMPTY) {
        fprintf(stderr, "%s\n", "Parameter is empty. Type 'tp0 -h' for help
            . Program terminated");
    }
    if (errorCode == ERR_INVALID_CHARS) {
        fprintf(stderr, "%s\n", "Argument has invalid characters. Type 'tp0
            -h' for help. Program terminated");
    }
    if (errorCode == ERR_INVALID_FORMAT) {
        fprintf(stderr, "%s\n", "Invalid arguments. Type 'tp0 -h' for help.
            Program terminated");
    }
    if (errorCode == ERR_INVALID_FILE_PATH) {
        fprintf(stderr, "%s\n", "File path is invalid. Type 'tp0 -h' for
            help. Program terminated");
    }
    if (errorCode == ERR_INVALID_PARAMETER) {
        fprintf(stderr, "%s\n", "Parameter input is invalid. Type 'tp0 -h'
            for help. Program terminated");
    }
}

```

```

    if (errorCode == ERR_INVALID_ACTION_TYPE) {
        fprintf(stderr, "%s\n", "Action is invalid. Type 'tp0 -h' for help.
            Program terminated");
    }
}

void process(parameters_t parameters){
    if(strcmp(parameters.action,"encode") == 0) {
        encode(parameters.input,parameters.path_to_input,parameters.output,
            parameters.path_to_output);
    }
    if(strcmp(parameters.action, "decode") == 0) {
        decode(parameters.input, parameters.path_to_input, parameters.
            output, parameters.path_to_output);
    }
}

int readInput(FILE* input, unsigned char* buff, int buffSize, int decode) {
    int i;
    int currentChar;
    for (i = 0; i < buffSize; ++i) {
        currentChar = fgetc(input);

        if (!ferror(input)) {
            if (currentChar != EOF) {
                if(!decode) buff[i] = currentChar;
                else{
                    if (currentChar == '\n') {
                        i--;
                        continue;
                    }
                    if(isValid(currentChar)==0){
                        fprintf(stderr, "Unable to decode:
                            invalid character. Program will
                            terminate.\n");
                        exit(1);
                    }
                    buff[i] = currentChar;
                }
            } else {
                return i;
            }
        } else {
            fprintf(stderr, "An error occurred when reading input.");
            exit(1);
        }
    }
    return buffSize;
}

void write_partial(unsigned char* processedOutput, FILE* output, char* path
) {
    if (output != NULL) {
        while (*processedOutput != '\0'){
            if (!(fprintf(output,"%c",*processedOutput))) {
                printf("%c\n", *processedOutput);
                fprintf(stderr, "Error when writing output to file %s\n",

```

```

        path);
        exit(1);
    }
    ++processedOutput;
}
} else {
    fprintf(stderr, "Unable to open or create output file %s\n", path);
    exit(1);
}
}

int main(int argc, char *argv[]) {
    int result = 0;
    parameters_t receivedParameters = getParams(argc, argv);
    process(receivedParameters);
    return result;
}

```

encode.h

```

#ifndef TP0_ENCODE_H
#define TP0_ENCODE_H

    void encode(FILE*,char*,FILE*,char*);
    int encodeChars(unsigned const char[], unsigned char[], int);
    int exceedsLineSize(int);

#endif //TP0_ENCODE_H

```

encode.c

```

#include <stdlib.h>
#include <stdio.h>
#include "encode.h"
#include "constants.h"

void encode(FILE*input, char*path, FILE * output, char * path_out) {
    unsigned char buffer[BLOCK_SIZE_INPUT_ENCODING];
    unsigned char encodedOutput[BLOCK_SIZE_OUTPUT_ENCODING];
    encodedOutput[4] = '\0';
    int length;
    int charsInLine = 0;

    if(path) input = fopen(path, "r");

    if(path_out) output = fopen(path_out, "w");

    if (!input) {
        fprintf(stderr, "Can't open the file %s\n", path);
        exit(1);
    }

    while (!feof(input)) {

        length = readInput(input, buffer, BLOCK_SIZE_INPUT_ENCODING, 0);

        if (length > 0) {
            encodeChars(buffer, encodedOutput, length);

```

```

        write_partial(encodedOutput, output, path_out);
        charsInLine += 4;
        if(exceedsLineSize(charsInLine) == 1){
            charsInLine = 0;
            encodedOutput[0] = '\n';
            encodedOutput[1] = '\0';
            write_partial(encodedOutput, output, path_out);
        }
    }
}

int exceedsLineSize(int charsInLine) {
    return (charsInLine == 76) ? 1:0;
}

int encodeChars(unsigned const char input[], unsigned char output[], int
length) {
    if (length == 3) {
        output[0] = encodingTable[input[0] >> 2];
        output[1] = encodingTable[((input[0] & 0x03) << 4) + (input[1] >>
4)];
        output[2] = encodingTable[((input[1] & 0x0f) << 2) + (input[2] >>
6)];
        output[3] = encodingTable[input[2] & 0x3F];
    } else if (length == 2) {
        output[0] = encodingTable[input[0] >> 2];
        output[1] = encodingTable[((input[0] & 0x03) << 4) + (input[1] >>
4)];
        output[2] = encodingTable[(input[1] & 0x0F) << 2];
        output[3] = '=';
    } else if (length == 1) {
        output[0] = encodingTable[input[0] >> 2];
        output[1] = encodingTable[(input[0] & 0x03) << 4];
        output[2] = '=';
        output[3] = '=';
    }
    return 4;
}

```

decode.h

```

#ifndef TP0_DECODE_H
#define TP0_DECODE_H

    void decode(FILE*,char*, FILE *, char *);
    int decodeChars(unsigned const char[], unsigned char[]);
    int isValid(unsigned char);

#endif //TP0_DECODE_H

```

decode.c

```

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "constants.h"

```



```

#include "decode.h"

void decode(FILE * input, char * path, FILE * output, char * path_out){
    unsigned char buffer[BLOCK_SIZE_INPUT_DECODING];
    unsigned char decoded_output[BLOCK_SIZE_INPUT_DECODING];
    decoded_output[4] = '\0';
    int chars_read = 0, length;

    if(path) input = fopen(path, "r");

    if(path_out) output = fopen(path_out, "w");

    if (!input) {
        fprintf(stderr, "Can't open the file %s\n", path);
        exit(1);
    }

    while(!feof(input)){
        length = readInput(input, buffer, BLOCK_SIZE_INPUT_DECODING, 1);
        chars_read += length;
        if (length && (length < BLOCK_SIZE_INPUT_DECODING)) {
            fprintf(stderr, "Can't finish decoding file %s: wrong
                length\n", path);
            break;
        }
        if(length > 0){
            decodeChars(buffer, decoded_output);
            write_partial(decoded_output, output, path_out);
        }
    }

    fclose(output);
}

int decodeChars(unsigned const char input[], unsigned char output[]){
    char decodedOutput[4] = {};
    decodedOutput[0] = decodingTable[input[0]];
    decodedOutput[1] = decodingTable[input[1]];
    decodedOutput[2] = decodingTable[input[2]];
    decodedOutput[3] = decodingTable[input[3]];

    output[0] = ((decodedOutput[0] << 2) | (decodedOutput[1] >> 4));
    if(decodedOutput[2] == EQUALS && decodedOutput[3] == EQUALS){
        output[1] = '\0';
        output[2] = '\0';
        return 1;
    }
    output[1] = (((decodedOutput[1] & 15) << 4) | ((decodedOutput[2] >>
        2) & 15));
    if(decodedOutput[3] == EQUALS){
        output[2] = '\0';
        return 2;
    }
    output[2] = (((decodedOutput[2] & 255) << 6) | ((decodedOutput[3])
        & 63));
    return 3;
}

```

```

int isValid(unsigned char currentChar){
    int i = 0;
    while (i < strlen((const char*)encodingTable)){
        if((currentChar == '=') || (encodingTable[i] == currentChar))
        {
            return 1;
        }
        i++;
    }
    return 0;
}

```

3.2. Codigo MIPS32

tp0.S

```

.file    1 "tp0.c"
.section .mdebug.abi32
.previous
.abicalls
.globl   encodingTable
.rdata
.align   2
.type    encodingTable, @object
.size    encodingTable, 65
encodingTable:
.ascii   "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123"
.ascii   "456789+/\000"
.globl   decodingTable
.align   2
.type    decodingTable, @object
.size    decodingTable, 256
decodingTable:
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -2
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1
.byte    -1

```

. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	-1
. byte	62
. byte	-1
. byte	-1
. byte	-1
. byte	63
. byte	52
. byte	53
. byte	54
. byte	55
. byte	56
. byte	57
. byte	58
. byte	59
. byte	60
. byte	61
. byte	-1
. byte	-1
. byte	-1
. byte	-3
. byte	-1
. byte	-1
. byte	-1
. byte	0
. byte	1
. byte	2
. byte	3
. byte	4
. byte	5
. byte	6
. byte	7
. byte	8
. byte	9
. byte	10
. byte	11
. byte	12
. byte	13
. byte	14
. byte	15
. byte	16

[illegible]

```

$LC0:      .align    2
           .ascii    "input\000"
           .align    2
$LC1:      .ascii    "output\000"
           .align    2
$LC2:      .ascii    "action\000"
           .align    2
$LC3:      .ascii    "help\000"
           .align    2
$LC4:      .ascii    "version\000"
           .data
           .align    2
           .type     long_options, @object
           .size     long_options, 96
long_options:
           .word     $LC0
           .word     1
           .word     0
           .word     105
           .word     $LC1
           .word     1
           .word     0
           .word     111
           .word     $LC2
           .word     1
           .word     0
           .word     97
           .word     $LC3
           .word     0
           .word     0
           .word     104
           .word     $LC4
           .word     0
           .word     0
           .word     86
           .word     0
           .word     0
           .word     0
           .word     0
           .rdata
           .align    2
$LC5:      .ascii    "-\000"
           .align    2
$LC6:      .ascii    "out\000"
           .align    2
$LC7:      .ascii    "in\000"
           .align    2
$LC8:      .ascii    "r\000"
           .align    2

```

```

$LC9:
    .ascii    "w\000"
    .text
    .align    2
    .globl    checkForPath
    .ent      checkForPath
checkForPath:
    .frame    $fp,48,$ra                # vars= 8, regs= 3/0, args= 16,
        extra= 8
    .mask     0xd0000000,-8
    .fmask     0x00000000,0
    .set       noreorder
    .cpload    $t9
    .set       reorder
    subu       $sp,$sp,48
    .cprestore 16
    sw         $ra,40($sp)
    sw         $fp,36($sp)
    sw         $gp,32($sp)
    move       $fp,$sp
    sw         $a0,48($fp)
    sw         $a1,52($fp)
    sw         $a2,56($fp)
    sw         $a3,60($fp)
    lw         $v0,48($fp)
    bne        $v0,$zero,$L18
    li         $v0,-1                    # 0xffffffffffffffff
    sw         $v0,28($fp)
    b          $L17
$L18:
    lw         $a0,48($fp)
    la         $t9,checkForInvalidCharacters
    jal        $ra,$t9
    move       $v1,$v0
    li         $v0,-1                    # 0xffffffffffffffff
    bne        $v1,$v0,$L19
    li         $v0,-2                    # 0xfffffffffffffffef
    sw         $v0,28($fp)
    b          $L17
$L19:
    lw         $a0,48($fp)
    la         $a1,$LC5
    la         $t9,strcmp
    jal        $ra,$t9
    bne        $v0,$zero,$L20
    lw         $a0,60($fp)
    la         $a1,$LC6
    la         $t9,strcmp
    jal        $ra,$t9
    bne        $v0,$zero,$L21
    la         $v0,___sF+88
    sw         $v0,52($fp)
$L21:
    lw         $a0,60($fp)
    la         $a1,$LC7
    la         $t9,strcmp
    jal        $ra,$t9
    bne        $v0,$zero,$L20

```



```

    la    $v0, __sF
    sw    $v0, 52($fp)
$L20:
    la    $v0, $LC8
    sw    $v0, 24($fp)
    lw    $a0, 60($fp)
    la    $a1, $LC6
    la    $t9, strcmp
    jal   $ra, $t9
    bne   $v0, $zero, $L23
    la    $v0, $LC9
    sw    $v0, 24($fp)
$L23:
    lw    $a0, 48($fp)
    lw    $a1, 24($fp)
    la    $t9, fopen
    jal   $ra, $t9
    sw    $v0, 52($fp)
    lw    $v0, 52($fp)
    bne   $v0, $zero, $L24
    li    $v0, -4
    sw    $v0, 28($fp)
    b     $L17
    # 0xfffffffffffffffc
$L24:
    lw    $v1, 56($fp)
    lw    $v0, 48($fp)
    sw    $v0, 0($v1)
    sw    $zero, 28($fp)
$L17:
    lw    $v0, 28($fp)
    move  $sp, $fp
    lw    $ra, 40($sp)
    lw    $fp, 36($sp)
    addu  $sp, $sp, 48
    j     $ra
    .end  checkForPath
    .size checkForPath, .-checkForPath
    .rdata
    .align 2
$LC10:
    .ascii "!@%^*|\000"
    .text
    .align 2
    .globl checkForInvalidCharacters
    .ent   checkForInvalidCharacters
checkForInvalidCharacters:
    .frame $fp, 56, $ra
    # vars= 16, regs= 3/0, args= 16,
    extra= 8
    .mask 0xd0000000, -8
    .fmask 0x00000000, 0
    .set  noreorder
    .cpload $t9
    .set  reorder
    subu  $sp, $sp, 56
    .cprestore 16
    sw    $ra, 48($sp)
    sw    $fp, 44($sp)
    sw    $gp, 40($sp)

```

```

        move    $fp,$sp
        sw      $a0,56($fp)
        lw      $v0,$LC10
        sw      $v0,24($fp)
        lhu     $v0,$LC10+4
        sh      $v0,28($fp)
        lbu     $v0,$LC10+6
        sb      $v0,30($fp)
        sw      $zero,32($fp)
$L26:
        addu    $a0,$fp,24
        la      $t9,strlen
        jal     $ra,$t9
        lw      $v1,32($fp)
        sltu    $v0,$v1,$v0
        bne     $v0,$zero,$L29
        b       $L27
$L29:
        lw      $v1,32($fp)
        addu    $v0,$fp,24
        addu    $v0,$v0,$v1
        lb      $v0,0($v0)
        lw      $a0,56($fp)
        move    $a1,$v0
        la      $t9, strchr
        jal     $ra,$t9
        beq     $v0,$zero,$L28
        li      $v0,-1
        sw      $v0,36($fp)
        b       $L25
        # 0xffffffffffffffff
$L28:
        lw      $v0,32($fp)
        addu    $v0,$v0,1
        sw      $v0,32($fp)
        b       $L26
$L27:
        sw      $zero,36($fp)
$L25:
        lw      $v0,36($fp)
        move    $sp,$fp
        lw      $ra,48($sp)
        lw      $fp,44($sp)
        addu    $sp,$sp,56
        j       $ra
        .end    checkForInvalidCharacters
        .size   checkForInvalidCharacters,.-checkForInvalidCharacters
        .rdata
        .align  2
$LC11:
        .ascii  "decode\000"
        .align  2
$LC12:
        .ascii  "encode\000"
        .text
        .align  2
        .globl  parseAction
        .ent    parseAction
parseAction:

```

```

.frame    $fp,48,$ra                                # vars= 8, regs= 3/0, args= 16,
        extra= 8
.mask     0xd0000000,-8
.fmask    0x00000000,0
.set      noreorder
.cpload   $t9
.set      reorder
subu      $sp,$sp,48
.cprestore 16
sw        $ra,40($sp)
sw        $fp,36($sp)
sw        $gp,32($sp)
move      $fp,$sp
sw        $a0,48($fp)
sw        $a1,52($fp)
lw        $v0,48($fp)
bne       $v0,$zero,$L32
li        $v0,-1                                     # 0xffffffffffffffff
sw        $v0,24($fp)
b         $L31
$L32:
lw        $a0,48($fp)
la        $a1,$LC11
la        $t9,streq
jal       $ra,$t9
bne       $v0,$zero,$L33
lw        $a0,48($fp)
la        $a1,$LC12
la        $t9,streq
jal       $ra,$t9
bne       $v0,$zero,$L33
li        $v0,-6                                     # 0xfffffffffffffffa
sw        $v0,24($fp)
b         $L31
$L33:
lw        $v1,52($fp)
lw        $v0,48($fp)
sw        $v0,0($v1)
sw        $zero,24($fp)
$L31:
lw        $v0,24($fp)
move      $sp,$fp
lw        $ra,40($sp)
lw        $fp,36($sp)
addu      $sp,$sp,48
j         $ra
.end      parseAction
.size     parseAction,.-parseAction
.rdata
.align    2
$LC13:
.ascii   "hVo:i:a:\000"
.align    2
$LC14:
.ascii   "No arguments provided for option -%c .\n\000"
.align    2
$LC15:
.ascii   "Unknown option '-%c' .\n\000"

```

```

    .align    2
$LC16:
    .ascii   "Unknown option '\\x%k'.\n\000"
    .text
    .align    2
    .globl   getParams
    .ent      getParams
getParams:
    .frame    $fp,88,$ra                # vars= 40, regs= 3/0, args= 24,
        extra= 8
    .mask     0xd0000000,-8
    .fmask    0x00000000,0
    .set      noreorder
    .cpload   $t9
    .set      reorder
    subu      $sp,$sp,88
    .cprestore 24
    sw        $ra,80($sp)
    sw        $fp,76($sp)
    sw        $gp,72($sp)
    move      $fp,$sp
    sw        $a0,88($fp)
    sw        $a1,92($fp)
    sw        $a2,96($fp)
    la        $v0,___sF+88
    sw        $v0,40($fp)
    sw        $zero,48($fp)
    la        $v0,___sF
    sw        $v0,44($fp)
    sw        $zero,52($fp)
    la        $v0,$LC12
    sw        $v0,56($fp)
$L35:
    sw        $zero,16($sp)
    lw        $a0,92($fp)
    lw        $a1,96($fp)
    la        $a2,$LC13
    la        $a3,long_options
    la        $t9,getopt_long
    jal       $ra,$t9
    sw        $v0,32($fp)
    lw        $v1,32($fp)
    li        $v0,-1                    # 0xffffffffffffffff
    bne       $v1,$v0,$L37
    b         $L36
$L37:
    lw        $v0,32($fp)
    addu      $v0,$v0,-63
    sw        $v0,64($fp)
    lw        $v1,64($fp)
    sltu      $v0,$v1,49
    beq       $v0,$zero,$L50
    lw        $v0,64($fp)
    sll       $v1,$v0,2
    la        $v0,$L51
    addu      $v0,$v1,$v0
    lw        $v0,0($v0)
    .cpadd    $v0

```


	lw	\$a1,40(\$fp)	
	move	\$a2,\$v0	
	la	\$a3,\$LC6	
	la	\$t9,checkForPath	
	jal	\$ra,\$t9	
	move	\$a0,\$v0	
	la	\$t9,validate	
	jal	\$ra,\$t9	
	b	\$L35	
\$L40:			
	addu	\$v0,\$fp,40	
	addu	\$v0,\$v0,12	
	lw	\$a0,optarg	
	lw	\$a1,44(\$fp)	
	move	\$a2,\$v0	
	la	\$a3,\$LC7	
	la	\$t9,checkForPath	
	jal	\$ra,\$t9	
	move	\$a0,\$v0	
	la	\$t9,validate	
	jal	\$ra,\$t9	
	b	\$L35	
\$L41:			
	addu	\$v0,\$fp,40	
	addu	\$v0,\$v0,16	
	lw	\$a0,optarg	
	move	\$a1,\$v0	
	la	\$t9,parseAction	
	jal	\$ra,\$t9	
	move	\$a0,\$v0	
	la	\$t9,validate	
	jal	\$ra,\$t9	
	b	\$L35	
\$L42:			
	la	\$t9,showVersion	
	jal	\$ra,\$t9	
	move	\$a0,\$zero	
	la	\$t9,exit	
	jal	\$ra,\$t9	
\$L43:			
	la	\$t9,showHelp	
	jal	\$ra,\$t9	
	move	\$a0,\$zero	
	la	\$t9,exit	
	jal	\$ra,\$t9	
\$L44:			
	lw	\$v1,optopt	
	li	\$v0,111	# 0x6f
	beq	\$v1,\$v0,\$L46	
	lw	\$v1,optopt	
	li	\$v0,105	# 0x69
	beq	\$v1,\$v0,\$L46	
	lw	\$v1,optopt	
	li	\$v0,97	# 0x61
	beq	\$v1,\$v0,\$L46	
	b	\$L45	
\$L46:			
	la	\$a0, __sF+176	

	la	\$a1,\$LC14	
	lw	\$a2,optopt	
	la	\$t9,fprintf	
	jal	\$ra,\$t9	
	b	\$L50	
\$L45:	lw	\$v1,_ctype_	
	lw	\$v0,optopt	
	addu	\$v0,\$v1,\$v0	
	addu	\$v0,\$v0,1	
	lbu	\$v0,0(\$v0)	
	andi	\$v0,\$v0,0x97	
	beq	\$v0,\$zero,\$L48	
	la	\$a0,___sF+176	
	la	\$a1,\$LC15	
	lw	\$a2,optopt	
	la	\$t9,fprintf	
	jal	\$ra,\$t9	
	b	\$L50	
\$L48:	la	\$a0,___sF+176	
	la	\$a1,\$LC16	
	lw	\$a2,optopt	
	la	\$t9,fprintf	
	jal	\$ra,\$t9	
\$L50:	la	\$t9,showHelp	
	jal	\$ra,\$t9	
	li	\$a0,1	# 0x1
	la	\$t9,exit	
	jal	\$ra,\$t9	
\$L36:	la	\$v0,___sF	
	bne	\$v0,\$zero,\$L52	
	la	\$t9,showHelp	
	jal	\$ra,\$t9	
	li	\$a0,1	# 0x1
	la	\$t9,exit	
	jal	\$ra,\$t9	
\$L52:	lw	\$v0,40(\$fp)	
	lw	\$v1,88(\$fp)	
	sw	\$v0,0(\$v1)	
	lw	\$v0,44(\$fp)	
	lw	\$v1,88(\$fp)	
	sw	\$v0,4(\$v1)	
	lw	\$v0,48(\$fp)	
	lw	\$v1,88(\$fp)	
	sw	\$v0,8(\$v1)	
	lw	\$v0,52(\$fp)	
	lw	\$v1,88(\$fp)	
	sw	\$v0,12(\$v1)	
	lw	\$v0,56(\$fp)	
	lw	\$v1,88(\$fp)	
	sw	\$v0,16(\$v1)	
	lw	\$v0,88(\$fp)	
	move	\$sp,\$fp	
	lw	\$ra,80(\$sp)	

```

        lw      $fp,76($sp)
        addu    $sp,$sp,88
        j       $ra
        .end    getParams
        .size   getParams,.-getParams
        .align  2
        .globl  validate
        .ent    validate
validate:
        .frame  $fp,40,$ra                # vars= 0, regs= 3/0, args= 16,
            extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cload  $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,40($fp)
        lw      $v0,40($fp)
        bgez    $v0,$L53
        lw      $a0,40($fp)
        la      $t9,showError
        jal     $ra,$t9
        li      $a0,1                    # 0x1
        la      $t9,exit
        jal     $ra,$t9
$L53:
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    validate
        .size   validate,.-validate
        .rdata
        .align  2
$LC17:
        .ascii  "%s\n\000"
        .align  2
$LC18:
        .ascii  "TP0 Organizacion de computadoras - HELP\000"
        .align  2
$LC19:
        .ascii  "Usage: \000"
        .align  2
$LC20:
        .ascii  "tp0 -h    Displays help and usage of the application\000"
        .align  2
$LC21:
        .ascii  "tp0 -V    Displays version of the application\000"
        .align  2
$LC22:
        .ascii  "tp0 Options:\000"

```



```

    .align    2
$LC23:
    .ascii    "-i --input          Set the location of the input file\000"
    .align    2
$LC24:
    .ascii    "-o --output          Set the location of the output file\000"
    .align    2
$LC25:
    .ascii    "-a --action          Program action: encode (default) or deco"
    .ascii    "de\000"
    .align    2
$LC26:
    .ascii    "Examples: \000"
    .align    2
$LC27:
    .ascii    "    tp0 -a encode -i ~/input -o ~/output\000"
    .align    2
$LC28:
    .ascii    "    tp0 -a decode\000"
    .text
    .align    2
    .globl    showHelp
    .ent      showHelp
showHelp:
    .frame    $fp,40,$ra                # vars= 0, regs= 3/0, args= 16,
        extra= 8
    .mask     0xd0000000,-8
    .fmask    0x00000000,0
    .set      noreorder
    .cplload  $t9
    .set      reorder
    subu      $sp,$sp,40
    .cprestore 16
    sw        $ra,32($sp)
    sw        $fp,28($sp)
    sw        $gp,24($sp)
    move      $fp,$sp
    la        $a0,$LC17
    la        $a1,$LC18
    la        $t9,printf
    jal       $ra,$t9
    la        $a0,$LC17
    la        $a1,$LC19
    la        $t9,printf
    jal       $ra,$t9
    la        $a0,$LC17
    la        $a1,$LC20
    la        $t9,printf
    jal       $ra,$t9
    la        $a0,$LC17
    la        $a1,$LC21
    la        $t9,printf
    jal       $ra,$t9
    la        $a0,$LC17
    la        $a1,$LC22
    la        $t9,printf
    jal       $ra,$t9
    la        $a0,$LC17

```

```

    la      $a1,$LC23
    la      $t9,printf
    jal     $ra,$t9
    la      $a0,$LC17
    la      $a1,$LC24
    la      $t9,printf
    jal     $ra,$t9
    la      $a0,$LC17
    la      $a1,$LC25
    la      $t9,printf
    jal     $ra,$t9
    la      $a0,$LC17
    la      $a1,$LC26
    la      $t9,printf
    jal     $ra,$t9
    la      $a0,$LC17
    la      $a1,$LC27
    la      $t9,printf
    jal     $ra,$t9
    la      $a0,$LC17
    la      $a1,$LC28
    la      $t9,printf
    jal     $ra,$t9
    move     $sp,$fp
    lw      $ra,32($sp)
    lw      $fp,28($sp)
    addu     $sp,$sp,40
    j        $ra
    .end     showHelp
    .size    showHelp, .-showHelp
    .rdata
    .align   2
$LC29:
    .ascii   "TP0 Organizacion de computadoras - VERSION: 1.0\000"
    .text
    .align   2
    .globl   showVersion
    .ent     showVersion
showVersion:
    .frame   $fp,40,$ra          # vars= 0, regs= 3/0, args= 16,
        extra= 8
    .mask    0xd0000000,-8
    .fmask   0x00000000,0
    .set     noreorder
    .cload   $t9
    .set     reorder
    subu     $sp,$sp,40
    .cprestore 16
    sw       $ra,32($sp)
    sw       $fp,28($sp)
    sw       $gp,24($sp)
    move     $fp,$sp
    la      $a0,$LC17
    la      $a1,$LC29
    la      $t9,printf
    jal     $ra,$t9
    move     $sp,$fp
    lw      $ra,32($sp)

```

```

        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    showVersion
        .size   showVersion,.-showVersion
        .rdata
        .align  2
$LC30:
        .ascii  "Parameter is empty. Type 'tp0 -h' for help. Program term"
        .ascii  "inated\000"
        .align  2
$LC31:
        .ascii  "Argument has invalid characters. Type 'tp0 -h' for help."
        .ascii  " Program terminated\000"
        .align  2
$LC32:
        .ascii  "Invalid arguments. Type 'tp0 -h' for help. Program termi"
        .ascii  "nated\000"
        .align  2
$LC33:
        .ascii  "File path is invalid. Type 'tp0 -h' for help. Program te"
        .ascii  "rminated\000"
        .align  2
$LC34:
        .ascii  "Parameter input is invalid. Type 'tp0 -h' for help. Prog"
        .ascii  "ram terminated\000"
        .align  2
$LC35:
        .ascii  "Action is invalid. Type 'tp0 -h' for help. Program termi"
        .ascii  "nated\000"
        .text
        .align  2
        .globl showError
        .ent    showError
showError:
        .frame  $fp,40,$ra                # vars= 0, regs= 3/0, args= 16,
            extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpld   $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,40($fp)
        lw      $v1,40($fp)
        li      $v0,-1                    # 0xffffffffffffffff
        bne     $v1,$v0,$L58
        la      $a0,___sF+176
        la      $a1,$LC17
        la      $a2,$LC30
        la      $t9,fprintf
        jal     $ra,$t9
$L58:

```

```

    lw      $v1,40($fp)
    li      $v0,-2                                # 0xfffffffffffffffe
    bne     $v1,$v0,$L59
    la      $a0,___sF+176
    la      $a1,$LC17
    la      $a2,$LC31
    la      $t9,fprintf
    jal     $ra,$t9
$L59:
    lw      $v1,40($fp)
    li      $v0,-3                                # 0xfffffffffffffffd
    bne     $v1,$v0,$L60
    la      $a0,___sF+176
    la      $a1,$LC17
    la      $a2,$LC32
    la      $t9,fprintf
    jal     $ra,$t9
$L60:
    lw      $v1,40($fp)
    li      $v0,-4                                # 0xfffffffffffffffc
    bne     $v1,$v0,$L61
    la      $a0,___sF+176
    la      $a1,$LC17
    la      $a2,$LC33
    la      $t9,fprintf
    jal     $ra,$t9
$L61:
    lw      $v1,40($fp)
    li      $v0,-5                                # 0xfffffffffffffffb
    bne     $v1,$v0,$L62
    la      $a0,___sF+176
    la      $a1,$LC17
    la      $a2,$LC34
    la      $t9,fprintf
    jal     $ra,$t9
$L62:
    lw      $v1,40($fp)
    li      $v0,-6                                # 0xfffffffffffffffa
    bne     $v1,$v0,$L57
    la      $a0,___sF+176
    la      $a1,$LC17
    la      $a2,$LC35
    la      $t9,fprintf
    jal     $ra,$t9
$L57:
    move     $sp,$fp
    lw      $ra,32($sp)
    lw      $fp,28($sp)
    addu     $sp,$sp,40
    j        $ra
    .end     showError
    .size    showError,.-showError
    .align   2
    .globl   process
    .ent     process
process:
    .frame   $fp,40,$ra                                # vars= 0, regs= 3/0, args= 16,
               extra= 8

```

```

        .mask      0xd0000000,-8
        .fmask     0x00000000,0
        .set       noreorder
        .cpload    $t9
        .set       reorder
        subu       $sp,$sp,40
        .cprestore 16
        sw         $ra,32($sp)
        sw         $fp,28($sp)
        sw         $gp,24($sp)
        move       $fp,$sp
        sw         $a0,40($fp)
        sw         $a1,44($fp)
        sw         $a2,48($fp)
        sw         $a3,52($fp)
        lw         $a0,56($fp)
        la         $a1,$LC12
        la         $t9,strcmp
        jal        $ra,$t9
        bne        $v0,$zero,$L65
        lw         $a0,44($fp)
        lw         $a1,52($fp)
        lw         $a2,40($fp)
        lw         $a3,48($fp)
        la         $t9,encode
        jal        $ra,$t9
$L65:
        lw         $a0,56($fp)
        la         $a1,$LC11
        la         $t9,strcmp
        jal        $ra,$t9
        bne        $v0,$zero,$L64
        lw         $a0,44($fp)
        lw         $a1,52($fp)
        lw         $a2,40($fp)
        lw         $a3,48($fp)
        la         $t9,decode
        jal        $ra,$t9
$L64:
        move       $sp,$fp
        lw         $ra,32($sp)
        lw         $fp,28($sp)
        addu       $sp,$sp,40
        j          $ra
        .end       process
        .size      process,.-process
        .rdata
        .align     2
$L63:
        .ascii     "Unable to decode: invalid character. Program will termin"
        .ascii     "ate.\n\000"
        .align     2
$L63:
        .ascii     "An error occurred when reading input.\000"
        .text
        .align     2
        .globl     readInput
        .ent       readInput

```

```

readInput:
    .frame    $fp,56,$ra                # vars= 16, regs= 3/0, args= 16,
        extra= 8
    .mask     0xd0000000,-8
    .fmask    0x00000000,0
    .set      noreorder
    .cpload   $t9
    .set      reorder
    subu      $sp,$sp,56
    .cprestore 16
    sw        $ra,48($sp)
    sw        $fp,44($sp)
    sw        $gp,40($sp)
    move      $fp,$sp
    sw        $a0,56($fp)
    sw        $a1,60($fp)
    sw        $a2,64($fp)
    sw        $a3,68($fp)
    sw        $zero,24($fp)
$L68:
    lw        $v0,24($fp)
    lw        $v1,64($fp)
    slt       $v0,$v0,$v1
    bne       $v0,$zero,$L71
    b         $L69
$L71:
    lw        $a0,56($fp)
    la        $t9,fgetc
    jal       $ra,$t9
    sw        $v0,28($fp)
    lw        $v0,56($fp)
    lhu       $v0,12($v0)
    srl       $v0,$v0,6
    andi      $v0,$v0,0x1
    bne       $v0,$zero,$L72
    lw        $v1,28($fp)
    li        $v0,-1                    # 0xffffffffffffffff
    beq       $v1,$v0,$L73
    lw        $v0,68($fp)
    bne       $v0,$zero,$L74
    lw        $v1,60($fp)
    lw        $v0,24($fp)
    addu      $v1,$v1,$v0
    lbu       $v0,28($fp)
    sb        $v0,0($v1)
    b         $L70
$L74:
    lw        $v1,28($fp)
    li        $v0,10                    # 0xa
    bne       $v1,$v0,$L76
    lw        $v0,24($fp)
    addu      $v0,$v0,-1
    sw        $v0,24($fp)
    b         $L70
$L76:
    lbu       $v0,28($fp)
    move      $a0,$v0
    la        $t9,isValid

```

```

jal    $ra,$t9
bne    $v0,$zero,$L77
la     $a0,___sF+176
la     $a1,$LC36
la     $t9,fprintf
jal    $ra,$t9
li     $a0,1                    # 0x1
la     $t9,exit
jal    $ra,$t9
$L77:
lw     $v1,60($fp)
lw     $v0,24($fp)
addu   $v1,$v1,$v0
lbu    $v0,28($fp)
sb     $v0,0($v1)
b      $L70
$L73:
lw     $v0,24($fp)
sw     $v0,32($fp)
b      $L67
$L72:
la     $a0,___sF+176
la     $a1,$LC37
la     $t9,fprintf
jal    $ra,$t9
li     $a0,1                    # 0x1
la     $t9,exit
jal    $ra,$t9
$L70:
lw     $v0,24($fp)
addu   $v0,$v0,1
sw     $v0,24($fp)
b      $L68
$L69:
lw     $v0,64($fp)
sw     $v0,32($fp)
$L67:
lw     $v0,32($fp)
move   $sp,$fp
lw     $ra,48($sp)
lw     $fp,44($sp)
addu   $sp,$sp,56
j      $ra
.end   readInput
.size  readInput,.-readInput
.rdata
.align 2
$LC38:
.ascii "%c\000"
.align 2
$LC39:
.ascii "%c\n\000"
.align 2
$LC40:
.ascii "Error when writing output to file %s\n\000"
.align 2
$LC41:
.ascii "Unable to open or create output file %s\n\000"

```

```

        .text
        .align 2
        .globl write_partial
        .ent   write_partial
write_partial:
        .frame $fp,40,$ra                # vars= 0, regs= 3/0, args= 16,
            extra= 8
        .mask 0xd0000000,-8
        .fmask 0x00000000,0
        .set   noreorder
        .cload $t9
        .set   reorder
        subu   $sp,$sp,40
        .cprestore 16
        sw     $ra,32($sp)
        sw     $fp,28($sp)
        sw     $gp,24($sp)
        move   $fp,$sp
        sw     $a0,40($fp)
        sw     $a1,44($fp)
        sw     $a2,48($fp)
        lw     $v0,44($fp)
        beq    $v0,$zero,$L81
$L82:
        lw     $v0,40($fp)
        lbu    $v0,0($v0)
        bne    $v0,$zero,$L84
        b      $L80
$L84:
        lw     $v0,40($fp)
        lbu    $v0,0($v0)
        lw     $a0,44($fp)
        la     $a1,$LC38
        move   $a2,$v0
        la     $t9,fprintf
        jal    $ra,$t9
        bne    $v0,$zero,$L85
        lw     $v0,40($fp)
        lbu    $v0,0($v0)
        la     $a0,$LC39
        move   $a1,$v0
        la     $t9,printf
        jal    $ra,$t9
        la     $a0,___sF+176
        la     $a1,$LC40
        lw     $a2,48($fp)
        la     $t9,fprintf
        jal    $ra,$t9
        li     $a0,1                    # 0x1
        la     $t9,exit
        jal    $ra,$t9
$L85:
        lw     $v0,40($fp)
        addu   $v0,$v0,1
        sw     $v0,40($fp)
        b      $L82
$L81:
        la     $a0,___sF+176

```



```

    la    $a1,$LC41
    lw    $a2,48($fp)
    la    $t9,fprintf
    jal   $ra,$t9
    li    $a0,1                # 0x1
    la    $t9,exit
    jal   $ra,$t9
$L80:
    move   $sp,$fp
    lw     $ra,32($sp)
    lw     $fp,28($sp)
    addu   $sp,$sp,40
    j      $ra
    .end    write_partial
    .size   write_partial,.-write_partial
    .align  2
    .globl  main
    .ent    main
main:
    .frame  $fp,80,$ra        # vars= 32, regs= 3/0, args= 24,
                               extra= 8
    .mask   0xd0000000,-8
    .fmask  0x00000000,0
    .set     noreorder
    .cload   $t9
    .set     reorder
    subu     $sp,$sp,80
    .cprestore 24
    sw       $ra,72($sp)
    sw       $fp,68($sp)
    sw       $gp,64($sp)
    move     $fp,$sp
    sw       $a0,80($fp)
    sw       $a1,84($fp)
    sw       $zero,32($fp)
    addu     $v0,$fp,40
    move     $a0,$v0
    lw       $a1,80($fp)
    lw       $a2,84($fp)
    la       $t9,getParams
    jal      $ra,$t9
    lw       $v0,56($fp)
    sw       $v0,16($sp)
    lw       $a0,40($fp)
    lw       $a1,44($fp)
    lw       $a2,48($fp)
    lw       $a3,52($fp)
    la       $t9,process
    jal      $ra,$t9
    lw       $v0,32($fp)
    move     $sp,$fp
    lw       $ra,72($sp)
    lw       $fp,68($sp)
    addu     $sp,$sp,80
    j        $ra
    .end     main
    .size    main,.-main
    .ident   "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"

```

encode.S

```

        .file      1 "encode.c"
        .section   .mdebug.abi32
        .previous
        .abicalls
        .rdata
        .align     2

$LC0:
        .ascii     "r\000"
        .align     2

$LC1:
        .ascii     "w\000"
        .align     2

$LC2:
        .ascii     "Can't open the file %s\n\000"
        .text
        .align     2
        .globl     encode
        .ent        encode

encode:
        .frame      $fp,64,$ra                # vars= 24, regs= 3/0, args= 16,
            extra= 8
        .mask       0xd0000000,-8
        .fmask      0x00000000,0
        .set        noreorder
        .cpload     $t9
        .set        reorder
        subu        $sp,$sp,64
        .cprestore  16
        sw          $ra,56($sp)
        sw          $fp,52($sp)
        sw          $gp,48($sp)
        move        $fp,$sp
        sw          $a0,64($fp)
        sw          $a1,68($fp)
        sw          $a2,72($fp)
        sw          $a3,76($fp)
        sb          $zero,36($fp)
        sw          $zero,44($fp)
        lw          $v0,68($fp)
        beq         $v0,$zero,$L18
        lw          $a0,68($fp)
        la          $a1,$LC0
        la          $t9,fopen
        jal         $ra,$t9
        sw          $v0,64($fp)

$L18:
        lw          $v0,76($fp)
        beq         $v0,$zero,$L19
        lw          $a0,76($fp)
        la          $a1,$LC1
        la          $t9,fopen
        jal         $ra,$t9
        sw          $v0,72($fp)

$L19:
        lw          $v0,64($fp)
        bne         $v0,$zero,$L20

```

	la	\$a0, __sF+176	
	la	\$a1, \$LC2	
	lw	\$a2, 68(\$fp)	
	la	\$t9, fprintf	
	jal	\$ra, \$t9	
	li	\$a0, 1	# 0x1
	la	\$t9, exit	
	jal	\$ra, \$t9	
\$L20:	.set	noreorder	
	nop		
	.set	reorder	
\$L21:	lw	\$v0, 64(\$fp)	
	lhu	\$v0, 12(\$v0)	
	srl	\$v0, \$v0, 5	
	andi	\$v0, \$v0, 0x1	
	beq	\$v0, \$zero, \$L23	
	b	\$L17	
\$L23:	lw	\$a0, 64(\$fp)	
	addu	\$a1, \$fp, 24	
	li	\$a2, 3	# 0x3
	move	\$a3, \$zero	
	la	\$t9, readInput	
	jal	\$ra, \$t9	
	sw	\$v0, 40(\$fp)	
	lw	\$v0, 40(\$fp)	
	blez	\$v0, \$L21	
	addu	\$v0, \$fp, 32	
	addu	\$a0, \$fp, 24	
	move	\$a1, \$v0	
	lw	\$a2, 40(\$fp)	
	la	\$t9, encodeChars	
	jal	\$ra, \$t9	
	addu	\$v0, \$fp, 32	
	move	\$a0, \$v0	
	lw	\$a1, 72(\$fp)	
	lw	\$a2, 76(\$fp)	
	la	\$t9, write_partial	
	jal	\$ra, \$t9	
	lw	\$v0, 44(\$fp)	
	addu	\$v0, \$v0, 4	
	sw	\$v0, 44(\$fp)	
	lw	\$a0, 44(\$fp)	
	la	\$t9, exceedsLineSize	
	jal	\$ra, \$t9	
	move	\$v1, \$v0	
	li	\$v0, 1	# 0x1
	bne	\$v1, \$v0, \$L21	
	sw	\$zero, 44(\$fp)	
	li	\$v0, 10	# 0xa
	sb	\$v0, 32(\$fp)	
	sb	\$zero, 33(\$fp)	
	addu	\$v0, \$fp, 32	
	move	\$a0, \$v0	
	lw	\$a1, 72(\$fp)	
	lw	\$a2, 76(\$fp)	

```

        la      $t9, write_partial
        jal     $ra, $t9
        b       $L21
$L17:
        move    $sp, $fp
        lw      $ra, 56($sp)
        lw      $fp, 52($sp)
        addu    $sp, $sp, 64
        j       $ra
        .end     encode
        .size    encode, .-encode
        .align   2
        .globl   exceedsLineSize
        .ent     exceedsLineSize
exceedsLineSize:
        .frame   $fp, 16, $ra                # vars= 0, regs= 2/0, args= 0,
            extra= 8
        .mask    0x50000000, -4
        .fmask   0x00000000, 0
        .set     noreorder
        .cload   $t9
        .set     reorder
        subu     $sp, $sp, 16
        .cprestore 0
        sw       $fp, 12($sp)
        sw       $gp, 8($sp)
        move     $fp, $sp
        sw       $a0, 16($fp)
        lw       $v0, 16($fp)
        xori     $v0, $v0, 0x4c
        sltu     $v0, $v0, 1
        move     $sp, $fp
        lw       $fp, 12($sp)
        addu     $sp, $sp, 16
        j       $ra
        .end     exceedsLineSize
        .size    exceedsLineSize, .-exceedsLineSize
        .align   2
        .globl   encodeChars
        .ent     encodeChars
encodeChars:
        .frame   $fp, 16, $ra                # vars= 0, regs= 2/0, args= 0,
            extra= 8
        .mask    0x50000000, -4
        .fmask   0x00000000, 0
        .set     noreorder
        .cload   $t9
        .set     reorder
        subu     $sp, $sp, 16
        .cprestore 0
        sw       $fp, 12($sp)
        sw       $gp, 8($sp)
        move     $fp, $sp
        sw       $a0, 16($fp)
        sw       $a1, 20($fp)
        sw       $a2, 24($fp)
        lw       $v1, 24($fp)
        li       $v0, 3                      # 0x3

```

	bne	\$v1,\$v0,\$L28	
	lw	\$v1,20(\$fp)	
	lw	\$v0,16(\$fp)	
	lbu	\$v0,0(\$v0)	
	srl	\$v0,\$v0,2	
	andi	\$v0,\$v0,0x00ff	
	lbu	\$v0,encodingTable(\$v0)	
	sb	\$v0,0(\$v1)	
	lw	\$v0,20(\$fp)	
	addu	\$a0,\$v0,1	
	lw	\$v0,16(\$fp)	
	lbu	\$v0,0(\$v0)	
	andi	\$v0,\$v0,0x3	
	sll	\$v1,\$v0,4	
	lw	\$v0,16(\$fp)	
	addu	\$v0,\$v0,1	
	lbu	\$v0,0(\$v0)	
	srl	\$v0,\$v0,4	
	andi	\$v0,\$v0,0x00ff	
	addu	\$v1,\$v1,\$v0	
	la	\$v0,encodingTable	
	addu	\$v0,\$v1,\$v0	
	lbu	\$v0,0(\$v0)	
	sb	\$v0,0(\$a0)	
	lw	\$v0,20(\$fp)	
	addu	\$a0,\$v0,2	
	lw	\$v0,16(\$fp)	
	addu	\$v0,\$v0,1	
	lbu	\$v0,0(\$v0)	
	andi	\$v0,\$v0,0xf	
	sll	\$v1,\$v0,2	
	lw	\$v0,16(\$fp)	
	addu	\$v0,\$v0,2	
	lbu	\$v0,0(\$v0)	
	srl	\$v0,\$v0,6	
	andi	\$v0,\$v0,0x00ff	
	addu	\$v1,\$v1,\$v0	
	la	\$v0,encodingTable	
	addu	\$v0,\$v1,\$v0	
	lbu	\$v0,0(\$v0)	
	sb	\$v0,0(\$a0)	
	lw	\$v0,20(\$fp)	
	addu	\$v1,\$v0,3	
	lw	\$v0,16(\$fp)	
	addu	\$v0,\$v0,2	
	lbu	\$v0,0(\$v0)	
	andi	\$v0,\$v0,0x3f	
	lbu	\$v0,encodingTable(\$v0)	
	sb	\$v0,0(\$v1)	
	b	\$L29	
\$L28:	lw	\$v1,24(\$fp)	
	li	\$v0,2	# 0x2
	bne	\$v1,\$v0,\$L30	
	lw	\$v1,20(\$fp)	
	lw	\$v0,16(\$fp)	
	lbu	\$v0,0(\$v0)	
	srl	\$v0,\$v0,2	

	andi	\$v0,\$v0,0x00ff	
	lbu	\$v0,encodingTable(\$v0)	
	sb	\$v0,0(\$v1)	
	lw	\$v0,20(\$fp)	
	addu	\$a0,\$v0,1	
	lw	\$v0,16(\$fp)	
	lbu	\$v0,0(\$v0)	
	andi	\$v0,\$v0,0x3	
	sll	\$v1,\$v0,4	
	lw	\$v0,16(\$fp)	
	addu	\$v0,\$v0,1	
	lbu	\$v0,0(\$v0)	
	srl	\$v0,\$v0,4	
	andi	\$v0,\$v0,0x00ff	
	addu	\$v1,\$v1,\$v0	
	la	\$v0,encodingTable	
	addu	\$v0,\$v1,\$v0	
	lbu	\$v0,0(\$v0)	
	sb	\$v0,0(\$a0)	
	lw	\$v0,20(\$fp)	
	addu	\$v1,\$v0,2	
	lw	\$v0,16(\$fp)	
	addu	\$v0,\$v0,1	
	lbu	\$v0,0(\$v0)	
	andi	\$v0,\$v0,0xf	
	sll	\$v0,\$v0,2	
	lbu	\$v0,encodingTable(\$v0)	
	sb	\$v0,0(\$v1)	
	lw	\$v0,20(\$fp)	
	addu	\$v1,\$v0,3	
	li	\$v0,61	# 0x3d
	sb	\$v0,0(\$v1)	
	b	\$L29	
\$L30:	lw	\$v1,24(\$fp)	
	li	\$v0,1	# 0x1
	bne	\$v1,\$v0,\$L29	
	lw	\$v1,20(\$fp)	
	lw	\$v0,16(\$fp)	
	lbu	\$v0,0(\$v0)	
	srl	\$v0,\$v0,2	
	andi	\$v0,\$v0,0x00ff	
	lbu	\$v0,encodingTable(\$v0)	
	sb	\$v0,0(\$v1)	
	lw	\$v0,20(\$fp)	
	addu	\$v1,\$v0,1	
	lw	\$v0,16(\$fp)	
	lbu	\$v0,0(\$v0)	
	andi	\$v0,\$v0,0x3	
	sll	\$v0,\$v0,4	
	lbu	\$v0,encodingTable(\$v0)	
	sb	\$v0,0(\$v1)	
	lw	\$v0,20(\$fp)	
	addu	\$v1,\$v0,2	
	li	\$v0,61	# 0x3d
	sb	\$v0,0(\$v1)	
	lw	\$v0,20(\$fp)	
	addu	\$v1,\$v0,3	

```

        li      $v0,61                      # 0x3d
        sb      $v0,0($v1)
$LC29:
        li      $v0,4                      # 0x4
        move    $sp,$fp
        lw      $fp,12($sp)
        addu    $sp,$sp,16
        j       $ra
        .end    encodeChars
        .size   encodeChars,.-encodeChars
        .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"

```

decode.S

```

        .file   1 "decode.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .rdata
        .align  2
$LC0:
        .ascii  "r\000"
        .align  2
$LC1:
        .ascii  "w\000"
        .align  2
$LC2:
        .ascii  "Can't open the file %s\n\000"
        .align  2
$LC3:
        .ascii  "Can't finish decoding file %s: wrong length\n\000"
        .text
        .align  2
        .globl  decode
        .ent    decode
decode:
        .frame   $fp,64,$ra                # vars= 24, regs= 3/0, args= 16,
            extra= 8
        .mask    0xd0000000,-8
        .fmask   0x00000000,0
        .set     noreorder
        .cload   $t9
        .set     reorder
        subu     $sp,$sp,64
        .cprestore 16
        sw      $ra,56($sp)
        sw      $fp,52($sp)
        sw      $gp,48($sp)
        move    $fp,$sp
        sw      $a0,64($fp)
        sw      $a1,68($fp)
        sw      $a2,72($fp)
        sw      $a3,76($fp)
        sb      $zero,36($fp)
        sw      $zero,40($fp)
        lw      $v0,68($fp)
        beq     $v0,$zero,$LC18
        lw      $a0,68($fp)
        la      $a1,$LC0

```

	la	\$t9, fopen	
	jal	\$ra, \$t9	
	sw	\$v0, 64(\$fp)	
\$L18:			
	lw	\$v0, 76(\$fp)	
	beq	\$v0, \$zero, \$L19	
	lw	\$a0, 76(\$fp)	
	la	\$a1, \$LC1	
	la	\$t9, fopen	
	jal	\$ra, \$t9	
	sw	\$v0, 72(\$fp)	
\$L19:			
	lw	\$v0, 64(\$fp)	
	bne	\$v0, \$zero, \$L20	
	la	\$a0, __sF+176	
	la	\$a1, \$LC2	
	lw	\$a2, 68(\$fp)	
	la	\$t9, fprintf	
	jal	\$ra, \$t9	
	li	\$a0, 1	# 0x1
	la	\$t9, exit	
	jal	\$ra, \$t9	
\$L20:			
	.set	noreorder	
	nop		
	.set	reorder	
\$L21:			
	lw	\$v0, 64(\$fp)	
	lhu	\$v0, 12(\$v0)	
	srl	\$v0, \$v0, 5	
	andi	\$v0, \$v0, 0x1	
	beq	\$v0, \$zero, \$L23	
	b	\$L22	
\$L23:			
	lw	\$a0, 64(\$fp)	
	addu	\$a1, \$fp, 24	
	li	\$a2, 4	# 0x4
	li	\$a3, 1	# 0x1
	la	\$t9, readInput	
	jal	\$ra, \$t9	
	sw	\$v0, 44(\$fp)	
	lw	\$v1, 40(\$fp)	
	lw	\$v0, 44(\$fp)	
	addu	\$v0, \$v1, \$v0	
	sw	\$v0, 40(\$fp)	
	lw	\$v0, 44(\$fp)	
	beq	\$v0, \$zero, \$L24	
	lw	\$v0, 44(\$fp)	
	slt	\$v0, \$v0, 4	
	beq	\$v0, \$zero, \$L24	
	la	\$a0, __sF+176	
	la	\$a1, \$LC3	
	lw	\$a2, 68(\$fp)	
	la	\$t9, fprintf	
	jal	\$ra, \$t9	
	b	\$L22	
\$L24:			
	lw	\$v0, 44(\$fp)	


```

        blez      $v0,$L21
        addu      $v0,$fp,32
        addu      $a0,$fp,24
        move      $a1,$v0
        la        $t9,decodeChars
        jal       $ra,$t9
        addu      $v0,$fp,32
        move      $a0,$v0
        lw        $a1,72($fp)
        lw        $a2,76($fp)
        la        $t9,write_partial
        jal       $ra,$t9
        b         $L21
$L22:
        lw        $a0,72($fp)
        la        $t9,fclose
        jal       $ra,$t9
        move      $sp,$fp
        lw        $ra,56($sp)
        lw        $fp,52($sp)
        addu      $sp,$sp,64
        j         $ra
        .end      decode
        .size     decode,.-decode
        .align    2
        .globl    decodeChars
        .ent      decodeChars
decodeChars:
        .frame    $fp,32,$ra                # vars= 16, regs= 2/0, args= 0,
            extra= 8
        .mask     0x50000000,-4
        .fmask    0x00000000,0
        .set      noreorder
        .cpload   $t9
        .set      reorder
        subu      $sp,$sp,32
        .cpstore  0
        sw        $fp,28($sp)
        sw        $gp,24($sp)
        move      $fp,$sp
        sw        $a0,32($fp)
        sw        $a1,36($fp)
        sw        $zero,8($fp)
        lw        $v0,32($fp)
        lbu       $v0,0($v0)
        lbu       $v0,decodingTable($v0)
        sb        $v0,8($fp)
        lw        $v0,32($fp)
        addu      $v0,$v0,1
        lbu       $v0,0($v0)
        lbu       $v0,decodingTable($v0)
        sb        $v0,9($fp)
        lw        $v0,32($fp)
        addu      $v0,$v0,2
        lbu       $v0,0($v0)
        lbu       $v0,decodingTable($v0)
        sb        $v0,10($fp)
        lw        $v0,32($fp)

```

	addu	\$v0,\$v0,3	
	lbu	\$v0,0(\$v0)	
	lbu	\$v0,decodingTable(\$v0)	
	sb	\$v0,11(\$fp)	
	lw	\$a0,36(\$fp)	
	lb	\$v0,8(\$fp)	
	sll	\$v1,\$v0,2	
	lb	\$v0,9(\$fp)	
	sra	\$v0,\$v0,4	
	or	\$v0,\$v1,\$v0	
	sb	\$v0,0(\$a0)	
	lb	\$v1,10(\$fp)	
	li	\$v0,-3	# 0xffffffffffffffffd
	bne	\$v1,\$v0,\$L27	
	lb	\$v1,11(\$fp)	
	li	\$v0,-3	# 0xffffffffffffffffd
	bne	\$v1,\$v0,\$L27	
	lw	\$v0,36(\$fp)	
	addu	\$v0,\$v0,1	
	sb	\$zero,0(\$v0)	
	lw	\$v0,36(\$fp)	
	addu	\$v0,\$v0,2	
	sb	\$zero,0(\$v0)	
	li	\$v0,1	# 0x1
	sw	\$v0,16(\$fp)	
	b	\$L26	
\$L27:	lw	\$v0,36(\$fp)	
	addu	\$a0,\$v0,1	
	lbu	\$v0,9(\$fp)	
	andi	\$v0,\$v0,0xf	
	sll	\$v1,\$v0,4	
	lb	\$v0,10(\$fp)	
	sra	\$v0,\$v0,2	
	andi	\$v0,\$v0,0xf	
	or	\$v0,\$v1,\$v0	
	sb	\$v0,0(\$a0)	
	lb	\$v1,11(\$fp)	
	li	\$v0,-3	# 0xffffffffffffffffd
	bne	\$v1,\$v0,\$L28	
	lw	\$v0,36(\$fp)	
	addu	\$v0,\$v0,2	
	sb	\$zero,0(\$v0)	
	li	\$v0,2	# 0x2
	sw	\$v0,16(\$fp)	
	b	\$L26	
\$L28:	lw	\$v0,36(\$fp)	
	addu	\$a0,\$v0,2	
	lb	\$v0,10(\$fp)	
	andi	\$v0,\$v0,0xff	
	sll	\$v1,\$v0,6	
	lbu	\$v0,11(\$fp)	
	andi	\$v0,\$v0,0x3f	
	or	\$v0,\$v1,\$v0	
	sb	\$v0,0(\$a0)	
	li	\$v0,3	# 0x3
	sw	\$v0,16(\$fp)	

```

$L26:
    lw      $v0,16($fp)
    move    $sp,$fp
    lw      $fp,28($sp)
    addu    $sp,$sp,32
    j       $ra
    .end    decodeChars
    .size   decodeChars,.-decodeChars
    .align  2
    .globl  isValid
    .ent    isValid
isValid:
    .frame  $fp,56,$ra                # vars= 16, regs= 3/0, args= 16,
        extra= 8
    .mask   0xd0000000,-8
    .fmask  0x00000000,0
    .set    noreorder
    .cload  $t9
    .set    reorder
    subu    $sp,$sp,56
    .cprestore 16
    sw      $ra,48($sp)
    sw      $fp,44($sp)
    sw      $gp,40($sp)
    move    $fp,$sp
    move    $v0,$a0
    sb      $v0,24($fp)
    sw      $zero,28($fp)

$L30:
    la      $a0,encodingTable
    la      $t9,strlen
    jal     $ra,$t9
    lw      $v1,28($fp)
    sltu    $v0,$v1,$v0
    bne     $v0,$zero,$L32
    b       $L31

$L32:
    lbu     $v1,24($fp)
    li      $v0,61                    # 0x3d
    beq     $v1,$v0,$L34
    lw      $v1,28($fp)
    la      $v0,encodingTable
    addu    $v0,$v1,$v0
    lbu     $v1,0($v0)
    lbu     $v0,24($fp)
    beq     $v1,$v0,$L34
    b       $L33

$L34:
    li      $v0,1                     # 0x1
    sw      $v0,32($fp)
    b       $L29

$L33:
    lw      $v0,28($fp)
    addu    $v0,$v0,1
    sw      $v0,28($fp)
    b       $L30

$L31:
    sw      $zero,32($fp)

```

```
$L29:
    lw      $v0,32($fp)
    move    $sp,$fp
    lw      $ra,48($sp)
    lw      $fp,44($sp)
    addu    $sp,$sp,56
    j       $ra
    .end    isValid
    .size   isValid,.-isValid
    .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

4. Conclusiones

La realizacion del presente trabajo practico nos sirvio para poder conocer y familiarizarnos con las herramientas y entornos con los que trabajaremos a lo largo de la materia, como por ejemplo GXemul y la plataforma NetBSD. Ademas, pudimos mejorar nuestros conocimientos en el lenguaje de programacion C y nuestra interaccion con entornos UNIX. Otro punto a notar, es que pudimos ver como obtener el codigo assembly de nuestro programa, y tener un primer acercamiento a un lenguaje de bajo nivel. Como nota adicional, nos parecio interesante aprender como se lleva a cabo la codificacion a base64 y viceversa, ya que es un protocolo que se utiliza frecuentemente para transferencia de contenido.

5. Referencias

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>

6. Anexo

6.1. Enunciado provisto por los profesores del curso

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>.