# Trabajo Practico N$^o$ 0 "Infraestructura Básica"

Nicolas Alvarez, *Padrón Nro. 93.503*
nicolasgalvarez91@gmail.com
Nicolás Fernández, *Padrón Nro. 88.599*
nflabo@gmail.com
Samanta Loiza, *Padrón Nro. 91.935*
samiloiza@gmail.com

1er. Cuatrimestre de 2015
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

# Índice

**Resumen**

El objetivo de este trabajo se basa en familiarizarse con las herramientas de software que se utilizará en los trabajos futuros. Para simular el entorno de desarrollo que se utilizará en los trabajos, se usó el programa GXemul, el cual nos permite trabajar sobre una máquina MIPS que corre el sistema operativo NetBSD. Para este trabajo se creó un programa el cual lee el contenido de uno o mas archivos y por cada línea, imprime el número de línea, seguido de la línea propiamente dicha.

# 1. Introducción

Para éste trabajo realizamos un programa en C el cual lee uno o más archivos de texto y los imprime por pantalla invirtiendo sus líneas.

Por último, utilizando la herramienta GXemul, llevamos nuestro código C a una máquina MIPS que corre el sistema operativo NetBSD y compilamos nuestro código sobre esta plataforma. De esta manera conseguimos el código Assembly generado por esta máquina.

# 2. Compilación y Ejecución del programa

Para compilar:

- Parados en la carpeta donde tenemos el fuente ejecutamos.

```
gcc -Wall -g Tp0.h -c Tp0.c
gcc -Wall -g Tp0.o Tp0Main.c -o Tp0
```

Para ejecutar:

- Parados en la carpeta donde tenemos el ejecutable generado por la compilación, el programa se invoca a través de línea de comandos de la siguiente manera:

```
./Tp0 [FILES]...
```

El menú de ayuda especifica las opciones disponibles al momento de invocar el programa:

```
Usage:
tp0 -h
tp0 -V
OPTIONS:
   -h, --help Print this information and quit.
   -V, --version Print version and quit
```

Ejemplo:

```
   ./Tp0 entrada1.txt entrada2.txt
    cat entrada1.txt | ./Tp0
```

# 3. Implementación

## 3.1. Desarrollo

Para el desarrollo del trabajo se realizó un programa escrito en lenguaje C. El usuario tendrá la posibilidad de ver un mensaje de ayuda.

- -h: Imprime el mensaje de ayuda.

Dependiendo de los parámetros introducidos el programa procederá a imprimir el texto, si se especificó el o los archivos de textos estos serán los que se impriman, caso contrario se procedera a imprimir el texto ingresado por la entrada estandar (stdin).

Finalemente el programa fue compilado en la máquina virtual proporcionada por el GXemul, para obtener el código MIPS32 generado por el compilador de dicha máquina.

## 3.2. Implementación en C

Implementamos diversas funciones para este trabajo, las cuales se encargan de:

- Checkear los argumentos recibidos:

  *int checkArguments(int,char*[])*

- Manejar los archivos que se ingresan por parametro (o el stdin en caso de no haber ingresado un archivo), para ser procesados:

  *int manejoArchivos()*

- Procesar un archivo, con el cual se lo lee y luego se guarda el archivo en forma invertida:

  *int procesarArchivo(FILE*,char*[],int)*

- Procesar la salida, ésta recibiendo las lineas ya invertidas procede a imprimir por pantalla y a liberar la memoria utilizada:

  *void procesarSalidaArchivo(char**, int,int )*

Para determinar cuántos serán los archivos a procesar se utiliza argc incluído en el main. Para poder guardar el archivo en memoria realizamos malloc para almacenar las letras de las líneas y para almacenar dichas líneas en un vector. En caso de que la memoria solicitada no sea lo suficientemente grande, procedemos a realizar un realloc.

# 4. Pruebas

Se utilizaron los siguientes archivos de prueba

- *test.txt*:

```
es la primera linea
y esta la segunda
la tercera es la vencida
```

- *largocorto.txt* (la linea que posee äf̈ue cortada para que pudiera representarse en la hoja):

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
bbbbbbbbb
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...
```

- *return.txt* (Este archivo se encuentra en el enunciado dado por la cátedra)
- *status.txt* (Este archivo se encuentra en el enunciado dado por la cátedra)

## 4.1. 1$^o$ Prueba

```
./Tp0 test.txt | ./Tp0

es la primera linea
y esta la segunda
la tercera es la vencida
```

## 4.2. 2$^o$ Prueba

```
 cat /usr/share/dict/words | md5sum
cbbcded3dc3b61ad50c4e36f79c37084 -
./Tp0 /usr/share/dict/words | ./Tp0 | md5sum
cbbcded3dc3b61ad50c4e36f79c37084 -
```

Podemos ver como en esta prueba se le aplica un hash a un conjunto de palabras, luego se aplica un ida y vuelta a dichas palabras y se aplica nuevamente el hash, dando en ambos casos el mismo valor de retorno por la función. También esta prueba nos permitió ver si se estaba solicitando memoria adecuadamente, ya que se procesa una gran cantidad de líneas.

### 4.3. 3º Prueba

```
./Tp0 vacio.txt
```

### 4.4. 4º Prueba

```
cat largocorto.txt | ./Tp0
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
bbbbbbbbb
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...
```

Con esta prueba podemos ver el caso donde el espacio solicitado para una linea no era suficiente y se debió pedir mas espacio para poder procesarla.

### 4.5. 5º Prueba

```
./Tp0 test.txt return.txt status.txt
la tercera es la vencida
y esta la segunda
Esta es la primera linea
of -1 is returned and errno is set to indicate the error.
Upon successful completion a value of 0 is returned. Otherwise, a value
file may not be changed
UF_IMMUTABLE
do not dump a file
UF_NODUMP
Description
Constant
The status information word st_flags has the following bits:
```

En este caso podemos ver el procesamiento simultáneo de 3 archivos, en el cual se procesa el primer archivo (*test.txt*), se lo invierte e imprime y luego se procesa el siguiente y así hasta haber invertido todos los archivos.

# 5. Conclusiones

El desarrollo de este trabajo nos permitió familiarizarnos con el uso del GXemul para lograr levantar una máquina virtual de MISP y poder trabajar en ella. Aprendimos como crear un túnel entre la máquina virtual del GXemul y una pc mediante el loopback, con el cual pudimos realizar traspasos de archivos de uno a otro. Con esto logramos codificar el programa en la computadora, pasarlo a la maquina virtual, compilarlo en ella y generar asi el código MIPS generado en dicha máquina. Pudimos notar la diferencia entre los códigos Assembly generados por el procesador de la computadora personal y los de la máquina virtual del GXemul. Utilizando la función md5sum pudimos corroborar que la transformación que realizabamos era correcta al obtener el mismo valor de retorno al aplicarlo sobre el texto antes de ser procesado por nuestra función *tac* y sobre el *tac* de la salida *tac* del texto anterior.

# 6. Codigo C

## 6.1. Tp0.h

```
/*
 * Tp0.h
 */

#ifndef Tp0_H_
#define Tp0_H_

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

#define BUF_TAM 100


/*Imprime la ayuda de parametros para el trabajo practico*/
void printHelp();
/*Se chequean los argumentos y en base a eso se imprime lo
    necesario */
int  checkArguments(int,char*[]);
/*Manejo global de los archivos, por cada uno se llama al
    metodo individual*/
int  manejoArchivos();
/*Metodo para procesamiento de cada archivo, recibe un File
    Descripto, los argumentos y numero de archivo*/
int procesarArchivo(FILE*,char*[],int);
/*Metodo que procesa la salida de las lineas de los archivos*/
void procesarSalidaArchivo(char**, int,int );

#endif /* Tp0_H_ */
```

## 6.2. Tp0Main.c

```
#include "Tp0.h"

int main(int argc, char* argv[]){
    if (!checkArguments(argc,argv))
        return 0;
    return manejoArchivos(argc,argv);
}
```

## 6.3. Tp0.c

```
#include "Tp0.h"

void printHelp() {
    char *help = "Usage:"
            "\ttp0 -h\n"
            "\ttp0 -V\n"
            "\ttp0 [file ...] \n"
```

```
            "Options: \n"
            " -V, --version \n\t Print version and quit. \n"
            " -h, --help \n\t Print this information and quit
                .\n"
            " Examples: \n"
            "  tp0 foo.txt bar.txt \n"
            "  tp0 gz.txt \n";

    printf("%s", help);
}
int checkArguments(int cantidadArgumentos, char* argumentos[])
    {
    int retorno = 1;
    if ((cantidadArgumentos == 2)
            && ((strcmp(argumentos[1], "-h") == 0)
                    || (strcmp(argumentos[1], "--help") == 0))
                        ) {
        printHelp();
        retorno = 0;
    } else if ((cantidadArgumentos == 2)
            && ((strcmp(argumentos[1], "-V") == 0)
                    || (strcmp(argumentos[1], "--version") ==
                        0))) {
        printf("Version 1.0\n");
        retorno = 0;
    }

    return retorno;
}

int procesarArchivo(FILE* fd, char* argumentos[], int
    numeroArchivo) {

    int j, estado = 0;
    char letra;
    long int bufTam = BUF_TAM;

    if (fd == NULL) {
        if (numeroArchivo != -1)
            fprintf(stderr, "Imposible abrir %s\n",
                    argumentos[numeroArchivo + 1]);
        else
            fprintf(stderr, "Error al leer de Stdin\n");
        return 1;
    }
    char** lines;
    int k = 0, finLineaSinN=0;
    estado = fread(&letra, 1, 1, fd); //Lee desde archivo 1
        elemento de 1 byte y lo guarda en letra.
    while (estado > 0) {
        char* line;
        j = 0;
        line = (char*) malloc(bufTam * sizeof(char));
```

```
        while ((estado > 0) && (letra != '\n')) {//mientras
            haya caracteres para leer y no haya llegado a fin
            de linea.

            if (j > bufTam) {   //realloc si es necesario
                bufTam += BUF_TAM;
                line = realloc(line, bufTam * sizeof(char));
            }

            line[j] = letra;
            j++;
            estado = fread(&letra, 1, 1, fd);
        }
        if (letra == '\n' || estado == 0) {
            if (k == 0) {
                lines = (char**) malloc(sizeof(char*));
            } else {
                lines = realloc(lines, sizeof(char*) * (k + 1)
                    );
            }
            line[j] = '\0';
            lines[k] = line;
            k++;
        } else if (letra != '\n' && estado == 0)  {
            finLineaSinN=1;
        }else{
            free(line);
        }
        estado = fread(&letra, 1, 1, fd);
    }
    fclose(fd);
    procesarSalidaArchivo(lines, k,finLineaSinN);
    return 0;
}

void procesarSalidaArchivo(char** lines, int k,int
    finLineaSinN) {
    int a;
    for (a = k - 1; a >= 0; a--) {
        if (a==0 && finLineaSinN)
            printf("%s", lines[a]);
        else
            printf("%s\n", lines[a]);
        free(lines[a]);
    }
    if (k != 0)
        free(lines);

}

int manejoArchivos(int cantidadParametros, char* argumentos[])
    {
    FILE* fd = NULL;
    int i, resultadoProcesamiento = 0;
```

```
    // Si no hay argumentos consideramos stdin como el unico
        archivo
    if (cantidadParametros == 1) {
        fd = stdin;
        resultadoProcesamiento = procesarArchivo(fd,
            argumentos, -1);
    } else {
        cantidadParametros -= 1;
        // Recorro los archivos para abrir y procesarlos
        for (i = 0; i < cantidadParametros &&
            resultadoProcesamiento == 0;
                i++) {
            fd = fopen(argumentos[i + 1], "r");
            resultadoProcesamiento = procesarArchivo(fd,
                argumentos, i);
        }
    }

    return resultadoProcesamiento;
}
```

# 7. Código MIPS32

## 7.1. Tp0.s

```
    .file   1 "Tp0.c"
    .section .mdebug.abi32
    .previous
    .abicalls
    .rdata
    .align  2
$LC0:
    .ascii  "Usage:\ttp0 -h \n"
    .ascii  "\ttp0 -V \n"
    .ascii  "\ttp0 [file ...] \n"
    .ascii  "Options: \n"
    .ascii  " -V, --version \n"
    .ascii  "\t Print version and quit. \n"
    .ascii  " -h, --help \n"
    .ascii  "\t Print this information and quit.\n"
    .ascii  " Examples: \n"
    .ascii  "  tp0 foo.txt bar.txt \n"
    .ascii  "  tp0 gz.txt \n\000"
    .align  2
$LC1:
    .ascii  "%s\000"
    .text
    .align  2
    .globl  printHelp
    .ent    printHelp
printHelp:
    .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
    la  $v0,$LC0
    sw  $v0,24($fp)
    la  $a0,$LC1
    lw  $a1,24($fp)
    la  $t9,printf
    jal $ra,$t9
    move    $sp,$fp
    lw  $ra,40($sp)
    lw  $fp,36($sp)
    addu    $sp,$sp,48
    j   $ra
```

```
    .end    printHelp
    .size   printHelp, .-printHelp
    .rdata
    .align  2
$LC2:
    .ascii  "-h\000"
    .align  2
$LC3:
    .ascii  "--help\000"
    .align  2
$LC4:
    .ascii  "-V\000"
    .align  2
$LC5:
    .ascii  "--version\000"
    .align  2
$LC6:
    .ascii  "Version 1.0\n\000"
    .text
    .align  2
    .globl  checkArguments
    .ent    checkArguments
checkArguments:
    .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)
    li  $v0,1           # 0x1
    sw  $v0,24($fp)
    lw  $v1,48($fp)
    li  $v0,2           # 0x2
    bne $v1,$v0,$L19
    lw  $v0,52($fp)
    addu    $v0,$v0,4
    lw  $a0,0($v0)
    la  $a1,$LC2
    la  $t9,strcmp
    jal $ra,$t9
    beq $v0,$zero,$L20
    lw  $v0,52($fp)
    addu    $v0,$v0,4
    lw  $a0,0($v0)
    la  $a1,$LC3
```

```
    la   $t9,strcmp
    jal  $ra,$t9
    bne  $v0,$zero,$L19
$L20:
    la   $t9,printHelp
    jal  $ra,$t9
    sw   $zero,24($fp)
    b    $L21
$L19:
    lw   $v1,48($fp)
    li   $v0,2               # 0x2
    bne  $v1,$v0,$L21
    lw   $v0,52($fp)
    addu     $v0,$v0,4
    lw   $a0,0($v0)
    la   $a1,$LC4
    la   $t9,strcmp
    jal  $ra,$t9
    beq  $v0,$zero,$L23
    lw   $v0,52($fp)
    addu     $v0,$v0,4
    lw   $a0,0($v0)
    la   $a1,$LC5
    la   $t9,strcmp
    jal  $ra,$t9
    bne  $v0,$zero,$L21
$L23:
    la   $a0,$LC6
    la   $t9,printf
    jal  $ra,$t9
    sw   $zero,24($fp)
$L21:
    lw   $v0,24($fp)
    move     $sp,$fp
    lw   $ra,40($sp)
    lw   $fp,36($sp)
    addu     $sp,$sp,48
    j    $ra
    .end     checkArguments
    .size    checkArguments, .-checkArguments
    .rdata
    .align   2
$LC7:
    .ascii   "Imposible abrir %s\n\000"
    .align   2
$LC8:
    .ascii   "Error al leer de Stdin\n\000"
    .text
    .align   2
    .globl   procesarArchivo
    .ent     procesarArchivo
procesarArchivo:
    .frame   $fp,80,$ra      # vars= 40, regs= 3/0, args= 16,
        extra= 8
```

```
    .mask   0xd0000000,-8
    .fmask  0x00000000,0
    .set    noreorder
    .cpload $t9
    .set    reorder
    subu    $sp,$sp,80
    .cprestore 16
    sw  $ra,72($sp)
    sw  $fp,68($sp)
    sw  $gp,64($sp)
    move    $fp,$sp
    sw  $a0,80($fp)
    sw  $a1,84($fp)
    sw  $a2,88($fp)
    sw  $zero,28($fp)
    li  $v0,100         # 0x64
    sw  $v0,36($fp)
    lw  $v0,80($fp)
    bne $v0,$zero,$L25
    lw  $v1,88($fp)
    li  $v0,-1          # 0xffffffffffffffff
    beq $v1,$v0,$L26
    lw  $v0,88($fp)
    sll $v1,$v0,2
    lw  $v0,84($fp)
    addu    $v0,$v1,$v0
    addu    $v0,$v0,4
    la  $a0,__sF+176
    la  $a1,$LC7
    lw  $a2,0($v0)
    la  $t9,fprintf
    jal $ra,$t9
    b   $L27
$L26:
    la  $a0,__sF+176
    la  $a1,$LC8
    la  $t9,fprintf
    jal $ra,$t9
$L27:
    li  $v0,1           # 0x1
    sw  $v0,56($fp)
    b   $L24
$L25:
    sw  $zero,44($fp)
    sw  $zero,48($fp)
    addu    $v0,$fp,32
    move    $a0,$v0
    li  $a1,1           # 0x1
    li  $a2,1           # 0x1
    lw  $a3,80($fp)
    la  $t9,fread
    jal $ra,$t9
    sw  $v0,28($fp)
$L28:
```

```
    lw   $v0,28($fp)
    bgtz     $v0,$L30
    b    $L29
$L30:
    sw   $zero,24($fp)
    lw   $a0,36($fp)
    la   $t9,malloc
    jal  $ra,$t9
    sw   $v0,52($fp)
$L31:
    lw   $v0,28($fp)
    blez     $v0,$L32
    lb   $v1,32($fp)
    li   $v0,10              # 0xa
    bne  $v1,$v0,$L33
    b    $L32
$L33:
    lw   $v0,24($fp)
    lw   $v1,36($fp)
    slt  $v0,$v1,$v0
    beq  $v0,$zero,$L35
    lw   $v0,36($fp)
    addu     $v0,$v0,100
    sw   $v0,36($fp)
    lw   $a0,52($fp)
    lw   $a1,36($fp)
    la   $t9,realloc
    jal  $ra,$t9
    sw   $v0,52($fp)
$L35:
    lw   $v1,52($fp)
    lw   $v0,24($fp)
    addu     $v1,$v1,$v0
    lbu  $v0,32($fp)
    sb   $v0,0($v1)
    lw   $v0,24($fp)
    addu     $v0,$v0,1
    sw   $v0,24($fp)
    addu     $v0,$fp,32
    move     $a0,$v0
    li   $a1,1               # 0x1
    li   $a2,1               # 0x1
    lw   $a3,80($fp)
    la   $t9,fread
    jal  $ra,$t9
    sw   $v0,28($fp)
    b    $L31
$L32:
    lb   $v1,32($fp)
    li   $v0,10              # 0xa
    beq  $v1,$v0,$L37
    lw   $v0,28($fp)
    bne  $v0,$zero,$L36
$L37:
```

```
    lw   $v0,44($fp)
    bne  $v0,$zero,$L38
    li   $a0,4              # 0x4
    la   $t9,malloc
    jal  $ra,$t9
    sw   $v0,40($fp)
    b    $L39
$L38:
    lw   $v0,44($fp)
    sll  $v0,$v0,2
    addu     $v0,$v0,4
    lw   $a0,40($fp)
    move     $a1,$v0
    la   $t9,realloc
    jal  $ra,$t9
    sw   $v0,40($fp)
$L39:
    lw   $v1,52($fp)
    lw   $v0,24($fp)
    addu     $v0,$v1,$v0
    sb   $zero,0($v0)
    lw   $v0,44($fp)
    sll  $v1,$v0,2
    lw   $v0,40($fp)
    addu     $v1,$v1,$v0
    lw   $v0,52($fp)
    sw   $v0,0($v1)
    lw   $v0,44($fp)
    addu     $v0,$v0,1
    sw   $v0,44($fp)
    b    $L40
$L36:
    lb   $v1,32($fp)
    li   $v0,10             # 0xa
    beq  $v1,$v0,$L41
    lw   $v0,28($fp)
    bne  $v0,$zero,$L41
    li   $v0,1              # 0x1
    sw   $v0,48($fp)
    b    $L40
$L41:
    lw   $a0,52($fp)
    la   $t9,free
    jal  $ra,$t9
$L40:
    addu     $v0,$fp,32
    move     $a0,$v0
    li   $a1,1              # 0x1
    li   $a2,1              # 0x1
    lw   $a3,80($fp)
    la   $t9,fread
    jal  $ra,$t9
    sw   $v0,28($fp)
    b    $L28
```

```
$L29:
    lw   $a0,80($fp)
    la   $t9,fclose
    jal  $ra,$t9
    lw   $a0,40($fp)
    lw   $a1,44($fp)
    lw   $a2,48($fp)
    la   $t9,procesarSalidaArchivo
    jal  $ra,$t9
    sw   $zero,56($fp)
$L24:
    lw   $v0,56($fp)
    move     $sp,$fp
    lw   $ra,72($sp)
    lw   $fp,68($sp)
    addu     $sp,$sp,80
    j    $ra
    .end     procesarArchivo
    .size    procesarArchivo, .-procesarArchivo
    .rdata
    .align   2
$LC9:
    .ascii   "%s\n\000"
    .text
    .align   2
    .globl   procesarSalidaArchivo
    .ent     procesarSalidaArchivo
procesarSalidaArchivo:
    .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)
    lw   $v0,52($fp)
    addu     $v0,$v0,-1
    sw   $v0,24($fp)
$L44:
    lw   $v0,24($fp)
    bgez     $v0,$L47
    b    $L45
$L47:
    lw   $v0,24($fp)
    bne  $v0,$zero,$L48
```

```
    lw   $v0,56($fp)
    beq  $v0,$zero,$L48
    lw   $v0,24($fp)
    sll  $v1,$v0,2
    lw   $v0,48($fp)
    addu     $v0,$v1,$v0
    la   $a0,$LC1
    lw   $a1,0($v0)
    la   $t9,printf
    jal  $ra,$t9
    b    $L49
$L48:
    lw   $v0,24($fp)
    sll  $v1,$v0,2
    lw   $v0,48($fp)
    addu     $v0,$v1,$v0
    la   $a0,$LC9
    lw   $a1,0($v0)
    la   $t9,printf
    jal  $ra,$t9
$L49:
    lw   $v0,24($fp)
    sll  $v1,$v0,2
    lw   $v0,48($fp)
    addu     $v0,$v1,$v0
    lw   $a0,0($v0)
    la   $t9,free
    jal  $ra,$t9
    lw   $v0,24($fp)
    addu     $v0,$v0,-1
    sw   $v0,24($fp)
    b    $L44
$L45:
    lw   $v0,52($fp)
    beq  $v0,$zero,$L43
    lw   $a0,48($fp)
    la   $t9,free
    jal  $ra,$t9
$L43:
    move     $sp,$fp
    lw   $ra,40($sp)
    lw   $fp,36($sp)
    addu     $sp,$sp,48
    j    $ra
    .end     procesarSalidaArchivo
    .size    procesarSalidaArchivo, .-procesarSalidaArchivo
    .rdata
    .align   2
$LC10:
    .ascii   "r\000"
    .text
    .align   2
    .globl   manejoArchivos
    .ent     manejoArchivos
```

```
manejoArchivos:
    .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  $zero,24($fp)
    sw  $zero,32($fp)
    lw  $v1,56($fp)
    li  $v0,1               # 0x1
    bne $v1,$v0,$L52
    la  $v0,__sF
    sw  $v0,24($fp)
    lw  $a0,24($fp)
    lw  $a1,60($fp)
    li  $a2,-1              # 0xffffffffffffffff
    la  $t9,procesarArchivo
    jal $ra,$t9
    sw  $v0,32($fp)
    b   $L53
$L52:
    lw  $v0,56($fp)
    addu    $v0,$v0,-1
    sw  $v0,56($fp)
    sw  $zero,28($fp)
$L54:
    lw  $v0,28($fp)
    lw  $v1,56($fp)
    slt $v0,$v0,$v1
    beq $v0,$zero,$L53
    lw  $v0,32($fp)
    bne $v0,$zero,$L53
    lw  $v0,28($fp)
    sll $v1,$v0,2
    lw  $v0,60($fp)
    addu    $v0,$v1,$v0
    addu    $v0,$v0,4
    lw  $a0,0($v0)
    la  $a1,$LC10
    la  $t9,fopen
    jal $ra,$t9
    sw  $v0,24($fp)
    lw  $a0,24($fp)
    lw  $a1,60($fp)
```

```
    lw   $a2,28($fp)
    la   $t9,procesarArchivo
    jal  $ra,$t9
    sw   $v0,32($fp)
    lw   $v0,28($fp)
    addu     $v0,$v0,1
    sw   $v0,28($fp)
    b    $L54
$L53:
    lw   $v0,32($fp)
    move     $sp,$fp
    lw   $ra,48($sp)
    lw   $fp,44($sp)
    addu     $sp,$sp,56
    j    $ra
    .end     manejoArchivos
    .size    manejoArchivos, .-manejoArchivos
    .ident   "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

## 7.2. Tp0Main.s

```
    .file    1 "Tp0Main.c"
    .section .mdebug.abi32
    .previous
    .abicalls
    .text
    .align   2
    .globl   main
    .ent     main
main:
    .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   $a0,48($fp)
    lw   $a1,52($fp)
    la   $t9,checkArguments
    jal  $ra,$t9
    bne  $v0,$zero,$L18
    sw   $zero,24($fp)
    b    $L17
$L18:
```

```
    lw   $a0,48($fp)
    lw   $a1,52($fp)
    la   $t9,manejoArchivos
    jal  $ra,$t9
    sw   $v0,24($fp)
$L17:
    lw   $v0,24($fp)
    move     $sp,$fp
    lw   $ra,40($sp)
    lw   $fp,36($sp)
    addu     $sp,$sp,48
    j    $ra
    .end     main
    .size    main, .-main
    .ident   "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

# Referencias

[1] GXemul, http://gavare.se/gxemul/.

[2] Tac (Wikipedia), http://en.wikipedia.org/wiki/Tac_(Unix).