



**FACULTAD
DE INGENIERIA**
Universidad de Buenos Aires

**Asignatura: (66.20) Organización de
Computadoras**

Curso: 1 (Martes)

Grupo: 1

TP N° 1: Programación MIPS¹

Nombre y apellido	Padrón	Correo electrónico
Bauni, Chiara	102981	cbauni@fi.uba.ar
Bilbao, Manuel	102732	mbilbao@fi.uba.ar
Stroia, Lautaro	100901	lstroia@fi.uba.ar

Primer cuatrimestre 2021

¹Link a repositorio: <https://github.com/chiabauni/OrgaDeCompus-TP1>

Índice

1. Enunciado	3
1.1. Objetivos	3
1.2. Alcance	3
1.3. Requisitos	3
1.4. Descripción	3
1.4.1. Ejemplos	4
1.5. Implementación	5
1.6. Informe	5
1.7. Regresiones	5
1.8. Entrega de TPs	6
2. Documentación relevante al diseño e implementación del programa	6
3. Comando(s) para compilar el programa	6
4. Corridas de prueba	7
4.1. Comandos de ayuda	7
4.2. Comandos de versión	8
4.3. Pruebas	9
4.3.1. Prueba: leer información desde un archivo y guardar el resultado en otro . .	9
4.3.2. Prueba: información en un archivo, resultados por salida standard	10
4.3.3. Prueba: información por stdin, salida por archivo	11
4.3.4. Prueba: información por stdin, salida por stdout	11
5. Código fuente	12
5.1. utils.h	12
5.2. main.c	12
5.3. max_divisor.h	15
5.4. max_divisor.c	16
5.5. gcd.S	19
6. Diagramas de Stack Frame	22
7. Código MIPS por el compilador	23
7.1. main.s	23
7.2. max_divisor.s	57

1. Enunciado

1.1. Objetivos

Familiarizarse con el conjunto de instrucciones MIPS y el concepto de ABI, extendiendo un programa que resuelva el problema descrito a continuación.

1.2. Alcance

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

1.3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes, un informe confeccionado de acuerdo con lo que mencionaremos en la sección 6, y con una copia digital de los archivos fuente necesarios para compilar el trabajo.

El trabajo y eventuales reentregas deberán ser presentadas exclusivamente a través del campus según las fechas preestablecidas para el mismo en la planificación del curso presentada en la primera clase del cuatrimestre. Asimismo todas las devoluciones se realizarán en horario de clase, en persona, para lo cual deberán estar presentes todos los integrantes.

1.4. Descripción

El programa a desarrollar deberá procesar un stream de texto compuesto por una cantidad arbitrariamente grande de líneas de longitud arbitraria.

La entrada del programa estará compuesta por una secuencia de pares de números enteros. Luego de leerlos, deberá calcular e imprimir el máximo común divisor (GCD) de cada par siguiendo los lineamientos indicados más abajo.

Por ejemplo, dado el siguiente flujo de entrada:

```
$ cat input.txt
60 45
90 9
17 13
```

Al ejecutar el programa la salida será:

```
$ tpl -i input.txt -o -
GCD(60, 45) = 15
GCD(90, 9) = 9
GCD(17, 13) = 1
```

1.4.1. Ejemplos

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

```
$ tp1 -h
Usage:
  tp1 -h
  tp1 -V
  tp1 -i in_file -o out_file
Options:
  -V, --version      Print version and quit.
  -h, --help         Print this information and quit.
  -i, --input        Specify input stream/file, "-" for stdin.
  -o, --output       Specify output stream/file, "-" for stdout.
Examples:
  tp1 < in.txt > out.txt
  cat in.txt | tp1 -i - > out.txt
```

A continuación, ejecutamos algunas pruebas: primero, veamos qué sucede cuando el archivo de entrada está vacío:

```
$ ./tp1 -o salida.txt </dev/null
$ ls -l salida.txt
-rw-r--r-- 1 leandro leandro 0 Oct 20 20:14 salida.txt
```

Aquí puede verse que el programa se comporta según lo esperado, ya que cuando la entrada está vacía, la salida lo estará también.

Veamos qué ocurre al ingresar un archivo con una única línea, la cual contiene un único par de valores:

```
$ echo 1 1 | ./tp1 -o -
1
```

Lo mismo debería ocurrir si la entrada se encuentra alojada en el sistema de archivos:

```
$ echo 1 1 >entrada.txt
$ ./tp1 -i entrada.txt -o -
1
```

1.5. Implementación

El programa a desarrollar constará de una mezcla entre código MIPS32 y C, siendo la parte escrita en assembly la encargada de calcular los GCDs utilizando el algoritmo de Euclides. El formato de dicha función será: **void euclides(struct gcd*, size_t);**

En donde el primer parámetro, de tipo struct gcd*, es un puntero a un arreglo de estructuras que describen los números de la entrada, mientras que el segundo parámetro es la cantidad de elementos presentes en el arreglo.

```
struct gcd {
    int num_a; /* input */
    int num_b; /* input */
    int gcd_ab; /* output */
};
```

De esta manera, con una única invocación a euclides(), el programa podrá calcular y retornar todos los GCDs usando el arreglo pasado por parámetro.

1.6. Informe

El informe deberá incluir:

- 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 y MIPS.
- El código MIPS32 generado por el compilador.
- Este enunciado.

1.7. Regresiones

El programa deberá pasar todas las regresiones definidas en el código fuente suministrado en este TP 1:

```
$ make
cc -Wall -g -o regressions regressions.c gcd.c gcd.S
:
$ make test
./regressions
```

Asimismo deberá usarse el modo 1 del sistema operativo para manejo de acceso no alineado a memoria.

1.8. Entrega de TPs

La entrega de este trabajo deberá realizarse usando el campus virtual de la materia dentro del plazo de tiempo preestablecido. Asimismo, en todos los casos, estas presentaciones deberán ser realizadas durante los días martes. El feedback estará disponible de un martes hacia el otro, para lo cual deberán estar presentes los integrantes de cada grupo tal como ocurre durante modalidad presencial de cursada.

Por otro lado, la última fecha de entrega y presentación para este trabajo será el martes 18/5.

2. Documentación relevante al diseño e implementación del programa

El objetivo de este primer trabajo práctico es familiarizarse con el conjunto de instrucciones MIPS y el concepto de ABI, a través de la creación de un algoritmo que busca el Máximo Común Divisor (GCD) entre 2 números conocido como **Algoritmo de Euclides**.

Lo primero que decidimos hacer fue crear un programa, escrito completamente en lenguaje C, para leer un archivo o la entrada standard y de ese input parsear la información de sus líneas buscando el GCD entre los números leídos.

Luego, basándonos en el código de **gcd.c**, procedimos a crear el archivo **gcd.S** en Assembly MIPS32, respetando la ABI de MIPS. Con este archivo finalizado, procedimos a la compilación y ejecución del código en la máquina virtual **QEMU**, que corre un SO Debian con kernel Linux para asegurarnos de que todo funcionase como corresponde.

Por último, se realizaron varias pruebas: algunas provistas por la cátedra en el archivo **regressions.c** y otras pasándole un archivo con números vía entrada standard para asegurarnos que el programa funcione como corresponde.

3. Comando(s) para compilar el programa

En el repositorio podemos encontrar dos carpetas: Enunciado y Resolución.

En la carpeta Enunciado tenemos los archivos **gcd.c**, **gcd.h**, **regressions.c** y un Makefile provistos por la cátedra y el **gcd.S** completado por nosotros.

Para correr las pruebas del archivo provisto por la cátedra:

1. **make;**
2. **./regressions**

Por otro lado encontramos la carpeta Resolución en donde están: **main.c** y **max_divisor.c** creados por nosotros para el manejo de archivos y el mismo **gcd.S** de la carpeta Enunciado.

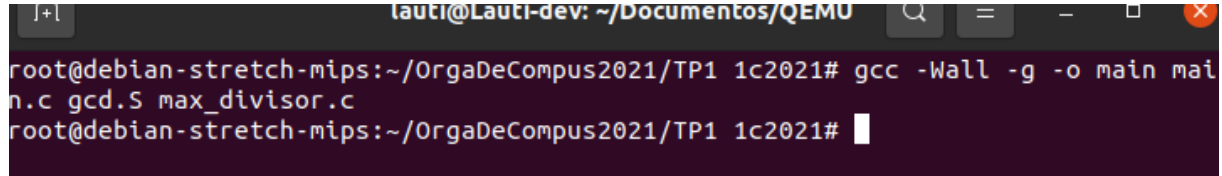
El comando para compilar el programa y obtener el ejecutable correctamente es:

1. **make;**
2. **./main -i tests/input.txt -o -**

También se puede correr el ejecutable con las distintas variaciones para ingresar el input y/o output.

4. Corridas de prueba

En esta sección mostraremos diversas pruebas realizadas para comprobar el correcto funcionamiento del programa. Para empezar, compilamos el archivo para obtener el ejecutable con la siguiente línea:



```
lauti@Lauti-dev: ~/Documentos/QEMU
root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021# gcc -Wall -g -o main main.c gcd.S max_divisor.c
root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021#
```

4.1. Comandos de ayuda

La primera prueba es para el comando de ayuda expresado con el flag **'-h'**, el cual muestra por salida standard al usuario las distintas opciones que tiene para ejecutar el programa.



```
lauti@Lauti-dev: ~/Documentos/QEMU
root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021# ./main -h
Comandos:
Uso:
ejecutable -h
ejecutable -V
ejecutable -i in_file -o out_file

Opciones:
-V, --version Imprime la version del programa y finaliza la ejecucion.
-h, --help Muestra esto por pantalla y finaliza la ejecucion.
-i, --input Especifica la ruta para el archivo de entrada, por default se usa stdin.
-o, --output Especifica la ruta para el archivo de salida, por default se usa stdout.

Ejemplos:
ejecutable < in.txt > out.txt
cat in.txt | ejecutable -i - > out.txt
root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021#
```

La forma alternativa a este comando, es utilizar el flag '**-help**'.

```
root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021# gcc -Wall -g -o main mairoot@
debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021# ./main --help
Comandos:
Uso:
ejecutable -h
ejecutable -V
ejecutable -i in_file -o out_file

Opciones:
-V, --version Imprime la version del programa y finaliza la ejecucion.
-h, --help Muestra esto por pantalla y finaliza la ejecucion.
-i, --input Especifica la ruta para el archivo de entrada, por default se usa stdin.
-o, --output Especifica la ruta para el archivo de salida, por default se usa stdout.

Ejemplos:
ejecutable < in.txt > out.txt
cat in.txt | ejecutable -i - > out.txtroot@debian-stretch-mips:~/OrgaDeCompus2021/TP1
1c2021# █
```

4.2. Comandos de versión

Similarmente, el usuario es capaz de obtener la versión del programa mediante el flag '**-V**'.

```
root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021# ./main -V
Organizacion de Computadoras TP1
1° cuatrimestre 2021
Entrega 1

Integrantes
Chiara Bauni 102981
Lautaro Stroia 100901
Manuel Bilbao 102732

Codigo disponible en : https://github.com/chiabauni/OrgaDeCompus-TP1root@debian-stret
ch-mips:~/OrgaDeCompus2021/TP1 1c2021# █
```

O también, puede utilizar el flag '**--version**'.

```
root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021# ./main --version
Organizacion de Computadoras TP1
1° cuatrimestre 2021
Entrega 1

Integrantes
Chiara Bauni 102981
Lautaro Stroia 100901
Manuel Bilbao 102732

Codigo disponible en : https://github.com/chiabauni/OrgaDeCompus-TP1root@debian-stret
ch-mips:~/OrgaDeCompus2021/TP1 1c2021# █
```


Y al abrir el archivo output.txt vemos lo siguiente:

[illegible]

4.3.2. Prueba: información en un archivo, resultados por salida standard

Ahora vamos a mostrar el funcionamiento del programa recibiendo el mismo archivo de entrada, `input.txt`, y devolviendo el resultado por `stdout`.

```
root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021# ./main -i input.txt -o -
GCD(1, 1) = 1
GCD(7, 1) = 1
GCD(13, 13) = 13
GCD(100, 10) = 10
GCD(100, 60) = 20
GCD(50, 75) = 25
GCD(60, 45) = 15
GCD(90, 9) = 9
GCD(17, 13) = 1
root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021#
```

4.3.3. Prueba: información por stdin, salida por archivo

Mostraremos el funcionamiento del programa escribiendo pares de números por entrada std, y guardando el resultado en un archivo de salida.

```
xtot@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021# ./main -i - -o output2.txt
1 5
4 3
100 25

root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021# vim output2.txt
```

[illegible]

4.3.4. Prueba: información por stdin, salida por stdout

Finalmente, mostramos por salida std el resultado del programa ingresando pares de números por entrada std.

```
root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021# ./main -i - -o -
175 6
190 3
45 4

GCD(175, 6) = 1
GCD(190, 3) = 1
GCD(45, 4) = 1
root@debian-stretch-mips:~/OrgaDeCompus2021/TP1 1c2021#
```

5. Código fuente

5.1. utils.h

Este archivo contiene la definición de constantes y funciones que van a ser utilizadas en el archivo principal.

```
#ifndef UTILS_H
#define UTILS_H

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <getopt.h>

#define MAX_MENSAJE 150
#define EOL '\n'
#define ERROR 1
#define EXITO 0

#define COMANDO_AYUDA_CORTO 'h'
#define COMANDO_VERSION_CORTO 'V'
#define COMANDO_AYUDA_LARGO "help"
#define COMANDO_VERSION_LARGO "version"
#define COMANDO_INPUT_CORTO 'i'
#define COMANDO_OUTPUT_CORTO 'o'
#define COMANDO_INPUT_LARGO "input"
#define COMANDO_OUTPUT_LARGO "output"

#define RUTA_AYUDA "comandos/help.txt"
#define RUTA_VERSION "comandos/version.txt"

#define MENSAJE_COMANDO_INVALIDO_ERROR "\n El comando usado es invalido ,\n use -h para ayuda.\n"
#define _MENSAJE_GCD_ERROR "\n No se pudo obtener el maximo comun divisor ,\n ocurrio un error.\n"
#define _MENSAJE_MEM_DINAMICA_ERROR "\n Problema asignando memoria dinamica.\n"
#define _MENSAJE_INPUT_ERROR "\n La cantidad de numeros ingresada es invalida.\n Solo se pueden ingresar 2 numeros por cada calculo de GCD.\n"
#define _MENSAJE_CONVERTIR_CHAR_ERROR "\n Error a la hora de convertir la\n linea de caracteres.\n"
#define _MENSAJE_LINEA_INVALIDA_ERROR "\n Problema leyendo el archivo.\n"

#endif //UTILS_H
```

5.2. main.c

Junto a **max_divisor.c** son los archivos principales del programa. Permiten manejar archivos de entrada y salida, tanto los standard como de archivos ingresados por stdin, y extraer información necesaria de su interior, para calcular el GCD.

```
#include "utils.h"
#include "max_divisor.h"
```

```

/*Pre: Recibe una ruta a un archivo en formato de string.
Pos: Muestra por stdin dicho archivo (similar al comando unix "cat".*/
int mostrar_en_pantalla(char* ruta);

/*Pre: Recibe una ruta a un archivo en formato de string.
Pos: Notifica por stderr que se tuvo un error con un archivo de dicha ruta.*/
void notificar_problema_ruta(char *ruta);

int main(int argc, char** argv){

    FILE* stream_entrada = NULL;
    FILE* stream_salida = NULL;
    int flag_divisor = -1;
    int long_index = 0;
    int opt = 0;
    char estandar[] = "-";

    static struct option long_options[] = {
        {COMANDO_VERSION_LARGO, no_argument, NULL, COMANDO_VERSION_CORTO },
        {COMANDO_AYUDA_LARGO, no_argument, NULL, COMANDO_AYUDA_CORTO },
        {COMANDO_INPUT_LARGO, required_argument, NULL, COMANDO_INPUT_CORTO },
        {COMANDO_OUTPUT_LARGO, required_argument, NULL, COMANDO_OUTPUT_CORTO },
        {NULL, 0, NULL, 0 }
    };

    while ( (opt = getopt_long(argc, argv, "Vhi:o:",
        long_options, &long_index )) != -1) {
    switch (opt) {
        case COMANDO_VERSION_CORTO :
            return mostrar_en_pantalla(RUTA_VERSION);
        case COMANDO_AYUDA_CORTO :
            return mostrar_en_pantalla(RUTA_AYUDA);
        case COMANDO_INPUT_CORTO :
            if (strcmp(optarg, estandar)) {
                stream_entrada = fopen(optarg, "r");
                if(stream_entrada == NULL){
                    notificar_problema_ruta(optarg);
                    if(stream_salida != NULL ){
                        //se pudo abrir el archivo de
                        //salida.
                        fclose(stream_salida);
                    }
                    return ERROR;
                }
            }

            break;
        case COMANDO_OUTPUT_CORTO :
            if (strcmp(optarg, estandar)) {
                stream_salida = fopen(optarg, "w");
                if(stream_salida == NULL){
                    notificar_problema_ruta(optarg);
                    if(stream_entrada != NULL ){
                        //se pudo abrir el archivo de
                        //entrada.

```

```

        fclose(stream_entrada);
    }
    return ERROR;
}

    }

    break;
default:
    perror(MENSAJE_COMANDO_INVALIDO_ERROR);
    return ERROR;
}
}

// Caso en que se usa la entrada/salida estandar.
if(stream_entrada == NULL ) stream_entrada = stdin;
if(stream_salida == NULL ) stream_salida = stdout;

    flag_divisor = procesar_archivos(stream_entrada , stream_salida);

    if(stream_salida != stdout) fclose(stream_salida);
    if(stream_entrada != stdin ) fclose(stream_entrada);

    if(flag_divisor == ERROR){
        perror(MENSAJE_GCD_ERROR);
        return ERROR;
    }

    return EXITO;
}

int mostrar_en_pantalla(char * ruta){
    FILE* archivo = fopen(ruta,"r");
    if (archivo == NULL){
        perror("\nNo se pudo abrir el archivo.\n");
        return ERROR;
    }

    int caracter = 1;

    while(caracter != EOF){
        caracter = fgetc(archivo);
        if(caracter != EOF) putc(caracter , stdout);
    }

    fclose(archivo);
    return EXITO;
}

void notificar_problema_ruta(char *ruta){
    char mensaje [MAX_MENSAJE];
    strcpy(mensaje , "\nEl archivo en la ruta: ");
    strcat(mensaje , ruta);
    strcat(mensaje , ". No se pudo abrir correctamente\n");
    perror(mensaje);
}

```

5.3. max_divisor.h

Contiene la definición de funciones utilizadas en **max_divisor.c** para manejo de archivos.

```
#ifndef _ORDENADOR_H
#define _ORDENADOR_H

#include "utils.h"
#include "gcd.h"

/*
Pre: Recibe un stream correctamente abierto en modo de lectura
     y otro en modo escritura.
Pos: En el caso que el stream de entrada tenga formato de lineas consecutivas
     de pares de numeros enteros separados por espacios. Calcula el maximo comun divisor y dicho resultado lo imprime en el
     stream de salida. Devolviendo el flag "EXITO". En caso que no se respete el
     formato se devuelve "FALLO".
*/
int procesar_archivos(FILE* entrada, FILE* salida);

/*
Pre: Recibe un file stream correctamente abierto en modo lectura.
Pos: Lee una linea de dicho stream, devolviendo por parametros la misma en
     forma de array de caracteres y el largo de la linea. En forma de retorno
     devuelve un valor indicando el resultado de la lectura:
     FIN_DE_ARCHIVO: En caso de encontrarse con un EOF.
     ERROR_LINEA_INVALIDA: En caso que se haya tenido que detener la ejecucion
     debido a un caracter invalido encontrado en el stream.
     ERROR_DEMEMORIA: En caso que se haya tenido que detener la ejecucion
     debido a un problema a la hora de reservar o asignar memoria dinamica.
     0: En caso de que el archivo continue.
*/
int leer_linea (FILE* stream, int *largo_linea, char** linea);

/*
Pre: Recibe un array de caracteres que contiene pares numeros enteros
     separados por un espacio(Esto es previamente validado) junto con su largo.
Pos: Devuelve por parametro el array de enteros y su largo.
*/
int pasar_a_enteros(char* linea, int largo_linea, int *enteros);

/*
Pre : Recibe un array de enteros, su largo y un stream de salida.
Pos: "Imprime" dicho array en el stream.
*/
void imprimir_salida(struct gcd *gcd, size_t n, FILE* salida);

/*
Pre: Recibe un caracter.
Pos: Devuelve TRUE si el caracter indica el fin de linea.
*/
bool es_fin_de_linea(char caracter);

/*
Pre: Recibe un caracter.
Pos: Devuelve TRUE si el caracter es considerado como numerico
*/
```

```

        (se considera los signos de + y - como numericos).
    */
    bool es_numerico(char caracter);

    /*
    Pre: Recibe un caracter
    Pos: Si el caracter es numerico, EOF, EOL o espacio devuelve TRUE.
        en otro caso devuelve FALSE. (dicho caracter no se deberia encontrar en el
        input).
    */
    bool es_caracter_invalido(char caracter);

#endif

```

5.4. max_divisor.c

Se encarga de procesar los archivos: reservando memoria para guardar los datos que se van leyendo linea por linea. Luego de leer una linea se pasa de un string a un array de dos enteros. Finalmente se imprime la salida en el archivo de salida.

```

#include "max_divisor.h"

#define FIN_DE_ARCHIVO 1
#define ERROR_LINEA_INVALIDA -1
#define ERROR_DEMEMORIA -2
#define ERROR_DEINPUT -3

int procesar_archivos(FILE* entrada, FILE* salida) {

    int lectura = 0;
    int enteros[2];
    int largo_linea = 0;
    char* linea = NULL;

    int largo_buffer = 20;
    size_t largo_arreglo = 0;
    struct gcd* arreglo_structs = malloc(sizeof(struct gcd) * largo_buffer);
    if((arreglo_structs) == NULL){
        perror(MENSAJE_MEMDINAMICA_ERROR);
        return ERROR_DEMEMORIA;
    }

    while(lectura == 0) {
        if((largo_arreglo) == (largo_buffer - 1)) {
            // Tengo que agrandar mi memoria
            largo_buffer += 10; //Voy agregando de a 10 lugares.
            arreglo_structs = realloc(arreglo_structs, sizeof(
                struct gcd) * largo_buffer); // Re ubico en la memoria.

            if((arreglo_structs) == NULL){
                perror(MENSAJE_MEMDINAMICA_ERROR);
                return ERROR_DEMEMORIA;
            }
        }
    }
}

```



```

        lectura = leer_linea(entrada, &largo_linea, &linea);

        if(lectura == ERROR_DEINPUT) {
            perror(MENSAJE_INPUT_ERROR);
            return ERROR;
        }

        if(lectura != ERROR_LINEA_INVALIDA) {

            if(pasar_a_enteros(linea, largo_linea, enteros)) {
                return ERROR;
            }

            arreglo_structs[largo_arreglo].num_a = enteros[0];
            arreglo_structs[largo_arreglo].num_b = enteros[1];
            arreglo_structs[largo_arreglo].gcd_ab = 0;
            largo_arreglo += 1;
        }
        free(linea);
    }

    if(lectura == ERROR_LINEA_INVALIDA) return ERROR;

    euclides(arreglo_structs, largo_arreglo);
    imprimir_salida(arreglo_structs, largo_arreglo, salida);
    free(arreglo_structs);
    return EXITO;
}

int leer_linea(FILE* stream, int *largo_linea, char** linea){
    int largo_buffer = 20;
    *linea = (char*) malloc(sizeof(char) * largo_buffer);
    // Asigno un lugar en memoria para el linea.

    if( (*linea) == NULL){
        return ERROR_DEMEMORIA;
    }

    (*largo_linea) = 0;
    int caracter = 1; // Un valor trivial

    while ( (caracter != EOL) && (caracter != EOF) ) {

        if( (*largo_linea) == (largo_buffer-1) ){
            // Tengo que agrandar mi memoria.
            largo_buffer +=10; // Voy agregando de a 10 lugares.
            (*linea) = (char*) realloc((*linea),
                sizeof(char) * largo_buffer);
            // Re ubico en la memoria.
            if( (*linea) == NULL){
                return ERROR_DEMEMORIA;
            }
        }

        if( ferror(stream) ){
            perror(MENSAJE_LINEA_INVALIDA_ERROR);

```

```

        return ERROR_LINEA_INVALIDA;
    }

    caracter = getc(stream); // Leo un caracter del stream.
    if( es_caracter_invalido(caracter) ){
        return ERROR_LINEA_INVALIDA; // Si lee un caracter que no
        //corresponde, devuelve linea invalida.
    }
    (*linea) [ (*largo_linea) ] = (char) caracter; //Lo guardo en el linea.
    (*largo_linea)++; // Incremento mi tope.
}

if(caracter == EOF || (*largo_linea) <=1){ // Siempre va a leer
//por lo menos un caracter, sea eof o fin de linea
    return FIN_DE_ARCHIVO;
}

return 0;
}

int pasar_a_enteros(char* linea, int largo_linea, int *enteros) {
    char temporal [largo_linea];
    char caracter = 'A';
    int largo_enteros = 0;

    int i = 0;
    int j = 0;

    while(i < largo_linea) {
        caracter = linea[i]; i++;
        if(largo_enteros >= 2) {
            perror(MENSAJE_INPUT_ERROR);
            return ERROR_DE_INPUT;
        } else if(es_numerico(caracter)) {
            temporal[j] = caracter; j++;
        } else if((caracter == '_' || es_fin_de_linea(caracter)) && j!=0) {
            temporal[j] = '\0';
            enteros[largo_enteros] = atoi(temporal);
            largo_enteros++;
            j = 0;
        }
    }
    return 0;
}

void imprimir_salida(struct gcd *gcd, size_t largo, FILE* salida) {
    if(largo == 0) return;

    for (int i = 0; i < (largo-1); i++) {
        fprintf(salida, "GCD(%i, %i) = %i\n", gcd[i].num_a,
            gcd[i].num_b, gcd[i].gcd_ab);
    }
}

bool es_fin_de_linea(char caracter) {

```

```

        return(caracter == EOL || caracter == (char) EOF);
}

bool es_numerico(char caracter) {
    return((caracter >= '0' && caracter <= '9')
        || caracter=='-' || caracter=='+');
}

bool es_caracter_invalido(char caracter) {
    return !(es_numerico(caracter) || es_fin_de_linea(caracter)
        || caracter == '_');
}

```

5.5. gcd.S

El **gcd.S** es el código MIPS encargado de buscar el GCD entre dos numeros que recibe por parámetro.

```

#include <sys/regdef.h>
    .abicalls
    .text
    .align 2
    .globl euclides
    .ent euclides

euclides:
    .frame fp, 36, ra                # Se crea el Stack Frame
    subu    sp, sp, 36

    .cprestore 28                    # Guarda gp en sp+28
    sw      ra, 32(sp)               # Se guardan los registros
    sw      fp, 24(sp)               # segun ABI
    sw      s0, 20(sp)
    move    fp, sp

    sw      a0, 36(fp)               # struct gcd* gcd
    sw      a1, 40(fp)               # size_t n
    sw      zero, 16(fp)             # size_t k = 0

    b       condicion_for

# Obtiene gcd[k].num_a y le aplica ABS
abs_num_a:
    lw      t1, 16(fp)               # t1 = k
    sll     t0, t1, 3                # t0 = t1 << 3 = k * 8
    sll     t1, t1, 2                # t1 = t1 << 2 = k * 4
    addu    t1, t1, t0               # t1 = t1 + t0 = 4*k + 8*k = 12*k

    lw      t0, 36(fp)               # t0 = gcd
    addu    s0, t0, t1               # s0 = t0 + t1 = gcd + 12*k = &(gcd[k])

    lw      t0, 0(s0)               # t0 = mem(s0) = *(&gcd[k]) = gcd[k].num_a

    # ABS(gcd[k].num_a)
    bgez    t0, abs_num_b           # if gcd[k].num_a >= 0 GOTO abs_num_b

```

```

        subu    t0, zero, t0                # t0 = -t0 = -gcd[k].num_a

# Obtiene gcd[k].num.b y le aplica ABS
abs_num_b:
        move    a0, t0                    # Prepara el primer argumento de euclides_

        lw      t0, 4(s0)                 # t0 = mem(s0+4) = *(&gcd[k] + 4) =
                                           # gcd[k].num_b

        # ABS(gcd[k].num.b)
        bgez    t0, llamada_euclides_     # if gcd[k].num_b >= 0 GOTO
                                           # llamada_euclides_

        subu    t0, zero, t0                # t0 = -t0 = -gcd[k].num_b

llamada_euclides_: # Llamada a euclides_ y ++k
        move    a1, t0                    # Prepara el segundo argumento de euclides_

        jal     euclides_

        # Guarda el resultado en memoria
        sw      v0, 8(s0)                 # gcd[k].gcd_ab = v0

        # Incrementa k
        lw      t0, 16(fp)                 # t0 = k
        addiu   t0, t0, 1                  # k++
        sw      t0, 16(fp)                 # Guarda k

condicion_for: # (k < n)
        lw      t1, 16(fp)                 # t1 = k
        lw      t0, 40(fp)                 # t0 = n
        sltu    t0, t1, t0                 # t0 = (k < n) ? 1 : 0
        bne     t0, zero, abs_num_a        # if (t0 != 0) GOTO abs_num_a
                                           # If (k < n) GOTO abs_num_a

        # Termina euclides
        move     sp, fp                    # Recupera los registros que se guardaron
        lw      ra, 32(sp)
        lw      fp, 24(sp)
        lw      s0, 20(sp)

        addiu    sp, sp, 36
        jr       ra                        # Vuelve al caller

        .end euclides

        .align 2
        .ent euclides_
euclides_:
        .frame   fp, 20, ra                # Se crea el Stack Frame
        subu     sp, sp, 20

        .cprestore 16                      # Se guardan los registros segun ABI
        sw       fp, 12(sp)

```

```

move    fp, sp

sw      a0, 20(fp)          # Guarda los parametros (a, b) en memoria
sw      a1, 24(fp)

# r2 = MAX(a, b)
move    t0, a1              # t0 = b
move    t1, a0              # t1 = a
slt     t2, t1, t0          # t2 = (t1 < t0) ? 1 : 0
                        # t2 = (a < b) ? 1 : 0

movz    t0, t1, t2          # if (t2 = 0) => t0 = t1
                        # If (a >= b) => t0 = a
sw      t0, 0(fp)           # mem(fp+0) = t0 (Guarda r2)

# r1 = MIN(a, b)
move    t0, a1              # t0 = b
move    t1, a0              # t1 = a
slt     t2, t0, t1          # t2 = (t0 < t1) ? 1 : 0
                        # t2 = (b < a) ? 1 : 0

movz    t0, t1, t2          # if (t2 = 0) => t0 = t1
                        # si b >= a => t0 = a
sw      t0, 4(fp)           # mem(fp+4) = t0 (Guarda r1)

while:
lw      t1, 0(fp)           # t1 = r2
lw      t0, 4(fp)           # t0 = r1

# If ((r0 = r2 % r1) == 0)
divu    t1, t0              # r2 / r1
mfhi    t2                  # t2 = r0 = r2 % r1
sw      t2, 8(fp)           # mem(fp+8) = t2 (Guarda r0)
beq     t2, zero, retorno_euclides_ # if (r0 == 0) GOTO
                        # retorno_euclides_ (break)

sw      t0, 0(fp)           # r2 = r1
sw      t2, 4(fp)           # r1 = r0
b       while              # while(1)

retorno_euclides_:
lw      v0, 4(fp)           # v0 = r1 (return r1)

move    sp, fp              # Recupera los registros guardados
lw      fp, 12(sp)
addiu   sp, sp, 20
jr      ra

.end euclides_

```

6. Diagramas de Stack Frame

A continuacion se muestran como quedan los stack frames, bajo determinadas circunstancias, de las funciones codificadas en Assembly en gcd.S

Referencias
Límite del stack frame
SRA
LTA
ABA

A la hora de llamar a la función euclides el stack frame se encuentra de la siguiente forma:

	...
	padding
SP + 40	n (size_t)
SP + 36	gcd (struct gcd*)
SP + 32	ra
SP + 28	gp
SP + 24	fp
SP + 20	s0
SP + 16	k (size_t)
SP + 12	
SP + 08	
SP + 04	
SP, FP	

En el siguiente diagrama se encuentra el stack frame de la función de euclides_:

	...
	padding
SP + 24	b (int)
SP + 20	a (int)
SP + 16	gp
SP + 12	fp
SP + 08	r0 (int)
SP + 04	r1 (int)
SP, FP	r2 (int)

7. Código MIPS por el compilador

Para finalizar, vamos a mostrar el código MIPS generado por el compilador GCC para los archivos main.c y max_divisor.c

7.1. main.s

```

.section .mdebug.abi32
.previous
.nan      legacy
.module fp=xx
.module nooddspreg
.abicalls
.text
$Ltext0:
.cfi_sections .debug_frame
.rdata
.align 2
$LC0:
.ascii "comandos/version.txt\000"
.align 2
$LC1:
.ascii "comandos/help.txt\000"
.align 2
$LC2:
.ascii "r\000"
.align 2
$LC3:
.ascii "w\000"
.align 2

```

```

$LC4:
    .ascii  "\012_El_comando_usado_es_invalido ,_use_-h_para_ayuda.\012"
    .ascii  "\000"
    .align  2
$LC5:
    .ascii  "Vhi:o:\000"
    .align  2
$LC6:
    .ascii  "\012_No_se_pudo_obtener_el_maximo_comun_divisor ,_ocurrio"
    .ascii  "_un_error.\012\000"
    .text
    .align  2
    .globl  main
$LFB2 = .
    .file 1 "main.c"
    .loc 1 13 0
    .cfi_startproc
    .set    nomips16
    .set    nomicromips
    .ent    main
    .type   main, @function
main:
    .frame  $fp,64,$31                # vars= 24, regs= 2/0, args= 24, gp= 8
    .mask   0xc0000000,-4
    .fmask   0x00000000,0
    .set     noreorder
    .cpload  $25
    .set     nomacro
    addiu    $sp,$sp,-64
    .cfi_def_cfa_offset 64
    sw       $31,60($sp)
    sw       $fp,56($sp)
    .cfi_offset 31, -4
    .cfi_offset 30, -8
    move     $fp,$sp
    .cfi_def_cfa_register 30
    .cprestore      24
    sw        $4,64($fp)
    sw        $5,68($fp)
    .loc 1 15 0
    sw        $0,32($fp)
    .loc 1 16 0
    sw        $0,36($fp)
    .loc 1 17 0
    li        $2,-1                    # 0xffffffffffffffff
    sw        $2,40($fp)
    .loc 1 18 0
    sw        $0,48($fp)
    .loc 1 19 0
    sw        $0,44($fp)
    .loc 1 20 0
    li        $2,11520                  # 0x2d00
    sh        $2,52($fp)
    .loc 1 31 0
    b         $L2
    nop

```



```

$L14:
    .loc 1 33 0
    lw      $2,44($fp)
    li      $3,104                # 0x68
    beq     $2,$3,$L4
    nop

    slt     $3,$2,105
    beq     $3,$0,$L5
    nop

    li      $3,86                 # 0x56
    beq     $2,$3,$L6
    nop

    b       $L3
    nop

$L5:
    li      $3,105                # 0x69
    beq     $2,$3,$L7
    nop

    li      $3,111                # 0x6f
    beq     $2,$3,$L8
    nop

    b       $L3
    nop

$L6:
    .loc 1 35 0
    lw      $2,%got($LC0)($28)
    addiu   $4,$2,%lo($LC0)
    lw      $2,%got(mostrar_en_pantalla)($28)
    move    $25,$2
    .reloc  1f,RMIPS_JALR,mostrar_en_pantalla
1:    jalr   $25
    nop

$LVLO = .
    lw      $28,24($fp)
    b       $L20
    nop

$L4:
    .loc 1 37 0
    lw      $2,%got($LC1)($28)
    addiu   $4,$2,%lo($LC1)
    lw      $2,%got(mostrar_en_pantalla)($28)
    move    $25,$2
    .reloc  1f,RMIPS_JALR,mostrar_en_pantalla
1:    jalr   $25
    nop

```

```

$LVL1 = .
        lw      $28,24($fp)
        b       $L20
        nop

$L7:
        .loc 1 39 0
        lw      $2,%got(optarg)($28)
        lw      $2,0($2)
        addiu   $3,$fp,52
        move    $5,$3
        move    $4,$2
        lw      $2,%call16(strcmp)($28)
        move    $25,$2
        .reloc 1f,R_MIPS_JALR,strcmp
1:      jalr    $25
        nop

$LVL2 = .
        lw      $28,24($fp)
        beq     $2,$0,$L2
        nop

        .loc 1 40 0
        lw      $2,%got(optarg)($28)
        lw      $3,0($2)
        lw      $2,%got($LC2)($28)
        addiu   $5,$2,%lo($LC2)
        move    $4,$3
        lw      $2,%call16(fopen)($28)
        move    $25,$2
        .reloc 1f,R_MIPS_JALR,fopen
1:      jalr    $25
        nop

$LVL3 = .
        lw      $28,24($fp)
        sw      $2,32($fp)
        .loc 1 41 0
        lw      $2,32($fp)
        bne     $2,$0,$L2
        nop

        .loc 1 42 0
        lw      $2,%got(optarg)($28)
        lw      $2,0($2)
        move    $4,$2
        lw      $2,%got(notificar_problema_ruta)($28)
        move    $25,$2
        .reloc 1f,R_MIPS_JALR,notificar_problema_ruta
1:      jalr    $25
        nop

$LVL4 = .
        lw      $28,24($fp)
        .loc 1 43 0

```

```

        lw      $2,36($fp)
        beq     $2,$0,$L11
        nop

        .loc 1 44 0
        lw      $4,36($fp)
        lw      $2,%call16(fclosen)( $28)
        move    $25,$2
        .reloc   1f,R_MIPS_JALR,fclosen
1:        jalr   $25
        nop

$LVL5 = .
        lw      $28,24($fp)
$L11:
        .loc 1 46 0
        li      $2,1                # 0x1
        b       $L20
        nop

$L8:
        .loc 1 51 0
        lw      $2,%got(optarg)( $28)
        lw      $2,0($2)
        addiu   $3,$fp,52
        move    $5,$3
        move    $4,$2
        lw      $2,%call16(strcmp)( $28)
        move    $25,$2
        .reloc   1f,R_MIPS_JALR,strcmp
1:        jalr   $25
        nop

$LVL6 = .
        lw      $28,24($fp)
        beq     $2,$0,$L2
        nop

        .loc 1 52 0
        lw      $2,%got(optarg)( $28)
        lw      $3,0($2)
        lw      $2,%got($LC3)( $28)
        addiu   $5,$2,%lo($LC3)
        move    $4,$3
        lw      $2,%call16(fopen)( $28)
        move    $25,$2
        .reloc   1f,R_MIPS_JALR,fopen
1:        jalr   $25
        nop

$LVL7 = .
        lw      $28,24($fp)
        sw      $2,36($fp)
        .loc 1 53 0
        lw      $2,36($fp)
        bne     $2,$0,$L2

```

```

        nop

        .loc 1 54 0
        lw      $2,%got(optarg)($28)
        lw      $2,0($2)
        move    $4,$2
        lw      $2,%got(notificar_problema_ruta)($28)
        move    $25,$2
        .reloc 1f,R_MIPS_JALR,notificar_problema_ruta
1:      jalr    $25
        nop

$VL8 = .
        lw      $28,24($fp)
        .loc 1 55 0
        lw      $2,32($fp)
        beq     $2,$0,$L13
        nop

        .loc 1 56 0
        lw      $4,32($fp)
        lw      $2,%call16(fclose)($28)
        move    $25,$2
        .reloc 1f,R_MIPS_JALR,fclose
1:      jalr    $25
        nop

$VL9 = .
        lw      $28,24($fp)
$L13:
        .loc 1 58 0
        li      $2,1                # 0x1
        b       $L20
        nop

$L3:
        .loc 1 63 0
        lw      $2,%got($LC4)($28)
        addiu   $4,$2,%lo($LC4)
        lw      $2,%call16(perror)($28)
        move    $25,$2
        .reloc 1f,R_MIPS_JALR,pererr
1:      jalr    $25
        nop

$VL10 = .
        lw      $28,24($fp)
        .loc 1 64 0
        li      $2,1                # 0x1
        b       $L20
        nop

$L2:
        .loc 1 31 0
        addiu   $2,$fp,48
        sw      $2,16($sp)

```

```

        lw      $2,%got(long_options.2593)($28)
        addiu   $7,$2,%lo(long_options.2593)
        lw      $2,%got($LC5)($28)
        addiu   $6,$2,%lo($LC5)
        lw      $5,68($fp)
        lw      $4,64($fp)
        lw      $2,%call16(getopt_long)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,getopt_long
1:        jalr   $25
        nop

$LVL11 = .
        lw      $28,24($fp)
        sw      $2,44($fp)
        lw      $3,44($fp)
        li      $2,-1                # 0xffffffffffffffff
        bne     $3,$2,$L14
        nop

        .loc 1 69 0
        lw      $2,32($fp)
        bne     $2,$0,$L15
        nop

        .loc 1 69 0 is_stmt 0 discriminator 1
        lw      $2,%got(stdin)($28)
        lw      $2,0($2)
        sw      $2,32($fp)
$L15:
        .loc 1 70 0 is_stmt 1
        lw      $2,36($fp)
        bne     $2,$0,$L16
        nop

        .loc 1 70 0 is_stmt 0 discriminator 1
        lw      $2,%got(stdout)($28)
        lw      $2,0($2)
        sw      $2,36($fp)
$L16:
        .loc 1 72 0 is_stmt 1
        lw      $5,36($fp)
        lw      $4,32($fp)
        lw      $2,%call16(procesar_archivos)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,procesar_archivos
1:        jalr   $25
        nop

$LVL12 = .
        lw      $28,24($fp)
        sw      $2,40($fp)
        .loc 1 75 0
        lw      $2,%got(stdout)($28)
        lw      $2,0($2)
        lw      $3,36($fp)

```

```

        beq      $3,$2,$L17
        nop

        .loc 1 75 0 is_stmt 0 discriminator 1
        lw       $4,36($fp)
        lw       $2,%call16(fclose)($28)
        move     $25,$2
        .reloc   1f,R_MIPS_JALR,fclose
1:        jalr    $25
        nop

$LVL13 = .
        lw       $28,24($fp)
$L17:
        .loc 1 76 0 is_stmt 1
        lw       $2,%got(stdin)($28)
        lw       $2,0($2)
        lw       $3,32($fp)
        beq      $3,$2,$L18
        nop

        .loc 1 76 0 is_stmt 0 discriminator 1
        lw       $4,32($fp)
        lw       $2,%call16(fclose)($28)
        move     $25,$2
        .reloc   1f,R_MIPS_JALR,fclose
1:        jalr    $25
        nop

$LVL14 = .
        lw       $28,24($fp)
$L18:
        .loc 1 78 0 is_stmt 1
        lw       $3,40($fp)
        li       $2,1                # 0x1
        bne      $3,$2,$L19
        nop

        .loc 1 79 0
        lw       $2,%got($LC6)($28)
        addiu    $4,$2,%lo($LC6)
        lw       $2,%call16(perror)($28)
        move     $25,$2
        .reloc   1f,R_MIPS_JALR,peror
1:        jalr    $25
        nop

$LVL15 = .
        lw       $28,24($fp)
        .loc 1 80 0
        li       $2,1                # 0x1
        b        $L20
        nop

$L19:
        .loc 1 83 0

```

```

        move    $2,$0
$L20:
        .loc 1 84 0 discriminator 1
        move    $sp,$fp
        .cfi_def_cfa_register 29
        lw      $31,60($sp)
        lw      $fp,56($sp)
        addiu   $sp,$sp,64
        .cfi_restore 30
        .cfi_restore 31
        .cfi_def_cfa_offset 0
        jr      $31
        nop

        .set     macro
        .set     reorder
        .end     main
        .cfi_endproc

$LFE2:
        .size    main, .-main
        .rdata
        .align   2

$LC7:
        .ascii   "\012_No_se_pudo_abrir_el_archivo.\012\000"
        .text
        .align   2
        .globl   mostrar_en_pantalla

$LFB3 = .
        .loc 1 86 0
        .cfi_startproc
        .set     nomips16
        .set     nomicromips
        .ent     mostrar_en_pantalla
        .type    mostrar_en_pantalla, @function
mostrar_en_pantalla:
        .frame   $fp,40,$31           # vars= 8, regs= 2/0, args= 16, gp= 8
        .mask    0xc0000000,-4
        .fmask   0x00000000,0
        .set     noreorder
        .cpload  $25
        .set     nomacro
        addiu    $sp,$sp,-40
        .cfi_def_cfa_offset 40
        sw       $31,36($sp)
        sw       $fp,32($sp)
        .cfi_offset 31, -4
        .cfi_offset 30, -8
        move     $fp,$sp
        .cfi_def_cfa_register 30
        .cprestore 16
        sw       $4,40($fp)
        .loc 1 87 0
        lw       $2,%got($LC2)($28)
        addiu    $5,$2,%lo($LC2)
        lw       $4,40($fp)
        lw       $2,%call16(fopen)($28)

```

```

        move    $25,$2
        .reloc  1f,R_MIPS_JALR,fopen
1:       jalr   $25
        nop

$LVL16 = .
        lw      $28,16($fp)
        sw      $2,28($fp)
        .loc 1 88 0
        lw      $2,28($fp)
        bne     $2,$0,$L22
        nop

        .loc 1 89 0
        lw      $2,%got($LC7)($28)
        addiu   $4,$2,%lo($LC7)
        lw      $2,%call16(perror)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,pererr
1:       jalr   $25
        nop

$LVL17 = .
        lw      $28,16($fp)
        .loc 1 90 0
        li      $2,1                # 0x1
        b       $L23
        nop

$L22:
        .loc 1 93 0
        li      $2,1                # 0x1
        sw      $2,24($fp)
        .loc 1 95 0
        b       $L24
        nop

$L25:
        .loc 1 96 0
        lw      $4,28($fp)
        lw      $2,%call16(fgetc)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,fgetc
1:       jalr   $25
        nop

$LVL18 = .
        lw      $28,16($fp)
        sw      $2,24($fp)
        .loc 1 97 0
        lw      $3,24($fp)
        li      $2,-1                # 0xffffffffffffffff
        beq     $3,$2,$L24
        nop

        .loc 1 97 0 is_stmt 0 discriminator 1

```



```

        lw      $2,%got(stdout)($28)
        lw      $2,0($2)
        move    $5,$2
        lw      $4,24($fp)
        lw      $2,%call16(_IO_putc)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,_IO_putc
1:      jalr    $25
        nop

$LVL19 = .
        lw      $28,16($fp)
$L24:
        .loc 1 95 0 is_stmt 1
        lw      $3,24($fp)
        li      $2,-1                # 0xffffffffffffffff
        bne     $3,$2,$L25
        nop

        .loc 1 100 0
        lw      $4,28($fp)
        lw      $2,%call16(fclosen)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,fclosen
1:      jalr    $25
        nop

$LVL20 = .
        lw      $28,16($fp)
        .loc 1 101 0
        move    $2,$0
$L23:
        .loc 1 102 0
        move    $sp,$fp
        .cfi_def_cfa_register 29
        lw      $31,36($sp)
        lw      $fp,32($sp)
        addiu   $sp,$sp,40
        .cfi_restore 30
        .cfi_restore 31
        .cfi_def_cfa_offset 0
        jr      $31
        nop

        .set     macro
        .set     reorder
        .end     mostrar_en_pantalla
        .cfi_endproc

$LF3:
        .size    mostrar_en_pantalla,.-mostrar_en_pantalla
        .rdata
        .align   2

$LC8:
        .ascii   "\012El archivo en la ruta: \000"
        .align   2

$LC9:

```

```

        .ascii   ".\No se pudo abrir correctamente\000\000"
        .text
        .align   2
        .globl   notificar_problema_ruta
$LFB4 = .
        .loc 1 104 0
        .cfi_startproc
        .set     nomips16
        .set     nomicromips
        .ent     notificar_problema_ruta
        .type    notificar_problema_ruta, @function
notificar_problema_ruta:
        .frame   $fp,184,$31                # vars= 152, regs= 2/0, args= 16, gp= 8
        .mask    0xc0000000,-4
        .fmask   0x00000000,0
        .set     noreorder
        .cpload  $25
        .set     nomacro
        addiu    $sp,$sp,-184
        .cfi_def_cfa_offset 184
        sw      $31,180($sp)
        sw      $fp,176($sp)
        .cfi_offset 31, -4
        .cfi_offset 30, -8
        move    $fp,$sp
        .cfi_def_cfa_register 30
        .cprestore 16
        sw      $4,184($fp)
        .loc 1 106 0
        lw      $2,%got($LC8)($28)
        lw      $8,%lo($LC8)($2)
        addiu   $3,$2,%lo($LC8)
        lw      $7,4($3)
        addiu   $3,$2,%lo($LC8)
        lw      $6,8($3)
        addiu   $3,$2,%lo($LC8)
        lw      $5,12($3)
        addiu   $3,$2,%lo($LC8)
        lw      $4,16($3)
        addiu   $3,$2,%lo($LC8)
        lw      $3,20($3)
        sw      $8,24($fp)
        sw      $7,28($fp)
        sw      $6,32($fp)
        sw      $5,36($fp)
        sw      $4,40($fp)
        sw      $3,44($fp)
        addiu   $2,$2,%lo($LC8)
        lbu     $2,24($2)
        sb      $2,48($fp)
        .loc 1 107 0
        lw      $5,184($fp)
        addiu   $2,$fp,24
        move    $4,$2
        lw      $2,%call16(strcat)($28)
        move    $25,$2

```

```

        .reloc 1f,R_MIPS_JALR, strcat
1:      jalr    $25
        nop

$LVL21 = .
        lw      $28,16($fp)
        .loc 1 108 0
        addiu   $2,$fp,24
        move    $4,$2
        lw      $2,%call16(strlen)($28)
        move    $25,$2
        .reloc 1f,R_MIPS_JALR,strlen
1:      jalr    $25
        nop

$LVL22 = .
        lw      $28,16($fp)
        move    $3,$2
        addiu   $2,$fp,24
        addu    $3,$2,$3
        lw      $2,%got($LC9)($28)
        move    $4,$3
        addiu   $2,$2,%lo($LC9)
        li      $3,33                # 0x21
        move    $6,$3
        move    $5,$2
        lw      $2,%call16(memcpy)($28)
        move    $25,$2
        .reloc 1f,R_MIPS_JALR,memcpy
1:      jalr    $25
        nop

$LVL23 = .
        lw      $28,16($fp)
        .loc 1 109 0
        addiu   $2,$fp,24
        move    $4,$2
        lw      $2,%call16(perror)($28)
        move    $25,$2
        .reloc 1f,R_MIPS_JALR, perror
1:      jalr    $25
        nop

$LVL24 = .
        lw      $28,16($fp)
        .loc 1 110 0
        nop
        move    $sp,$fp
        .cfi_def_cfa_register 29
        lw      $31,180($sp)
        lw      $fp,176($sp)
        addiu   $sp,$sp,184
        .cfi_restore 30
        .cfi_restore 31
        .cfi_def_cfa_offset 0
        jr      $31

```

```

        nop

        .set      macro
        .set      reorder
        .end      notificar_problema_ruta
        .cfi_endproc

$LFE4:
        .size     notificar_problema_ruta , .-notificar_problema_ruta
        .rdata
        .align    2

$LC10:
        .ascii    "version\000"
        .align    2

$LC11:
        .ascii    "help\000"
        .align    2

$LC12:
        .ascii    "input\000"
        .align    2

$LC13:
        .ascii    "output\000"
        .section   .data.rel.local,"aw",@progbits
        .align    2
        .type     long_options.2593, @object
        .size     long_options.2593, 80
long_options.2593:
        .word     $LC10
        .word     0
        .word     0
        .word     86
        .word     $LC11
        .word     0
        .word     0
        .word     104
        .word     $LC12
        .word     1
        .word     0
        .word     105
        .word     $LC13
        .word     1
        .word     0
        .word     111
        .word     0
        .word     0
        .word     0
        .word     0
        .text

$Letext0:
        .file     2 "/usr/lib/gcc/mips-linux-gnu/6/include/stddef.h"
        .file     3 "/usr/include/mips-linux-gnu/bits/types.h"
        .file     4 "/usr/include/stdio.h"
        .file     5 "/usr/include/libio.h"
        .file     6 "/usr/include/mips-linux-gnu/bits/sys_errlist.h"
        .file     7 "/usr/include/getopt.h"
        .section   .debug_info,"",@progbits
$Ldebug_info0:

```

```

.4 byte 0x4dc
.2 byte 0x4
.4 byte $Ldebug_abbrev0
.byte 0x4
.uleb128 0x1
.4 byte $LASF81
.byte 0xc
.4 byte $LASF82
.4 byte $LASF83
.4 byte $Ltext0
.4 byte $Ltext0-$Ltext0
.4 byte $Ldebug_line0
.uleb128 0x2
.4 byte $LASF8
.byte 0x2
.byte 0xd8
.4 byte 0x30
.uleb128 0x3
.byte 0x4
.byte 0x7
.4 byte $LASF0
.uleb128 0x3
.byte 0x1
.byte 0x8
.4 byte $LASF1
.uleb128 0x3
.byte 0x2
.byte 0x7
.4 byte $LASF2
.uleb128 0x3
.byte 0x4
.byte 0x7
.4 byte $LASF3
.uleb128 0x3
.byte 0x1
.byte 0x6
.4 byte $LASF4
.uleb128 0x3
.byte 0x2
.byte 0x5
.4 byte $LASF5
.uleb128 0x4
.byte 0x4
.byte 0x5
.ascii "int\000"
.uleb128 0x3
.byte 0x8
.byte 0x5
.4 byte $LASF6
.uleb128 0x3
.byte 0x8
.byte 0x7
.4 byte $LASF7
.uleb128 0x2
.4 byte $LASF9
.byte 0x3

```

```
.byte    0x37
.4byte   0x61
.uleb128 0x2
.4byte   $LASF10
.byte    0x3
.byte    0x83
.4byte   0x85
.uleb128 0x3
.byte    0x4
.byte    0x5
.4byte   $LASF11
.uleb128 0x2
.4byte   $LASF12
.byte    0x3
.byte    0x84
.4byte   0x6f
.uleb128 0x3
.byte    0x4
.byte    0x7
.4byte   $LASF13
.uleb128 0x5
.byte    0x4
.uleb128 0x6
.byte    0x4
.4byte   0xa6
.uleb128 0x3
.byte    0x1
.byte    0x6
.4byte   $LASF14
.uleb128 0x7
.4byte   0xa6
.uleb128 0x2
.4byte   $LASF15
.byte    0x4
.byte    0x30
.4byte   0xbd
.uleb128 0x8
.4byte   $LASF45
.byte    0x98
.byte    0x5
.byte    0xf1
.4byte   0x23a
.uleb128 0x9
.4byte   $LASF16
.byte    0x5
.byte    0xf2
.4byte   0x5a
.byte    0
.uleb128 0x9
.4byte   $LASF17
.byte    0x5
.byte    0xf7
.4byte   0xa0
.byte    0x4
.uleb128 0x9
.4byte   $LASF18
```

```

.byte    0x5
.byte    0xf8
.4byte   0xa0
.byte    0x8
.uleb128 0x9
.4byte   $LASF19
.byte    0x5
.byte    0xf9
.4byte   0xa0
.byte    0xc
.uleb128 0x9
.4byte   $LASF20
.byte    0x5
.byte    0xfa
.4byte   0xa0
.byte    0x10
.uleb128 0x9
.4byte   $LASF21
.byte    0x5
.byte    0xfb
.4byte   0xa0
.byte    0x14
.uleb128 0x9
.4byte   $LASF22
.byte    0x5
.byte    0xfc
.4byte   0xa0
.byte    0x18
.uleb128 0x9
.4byte   $LASF23
.byte    0x5
.byte    0xfd
.4byte   0xa0
.byte    0x1c
.uleb128 0x9
.4byte   $LASF24
.byte    0x5
.byte    0xfe
.4byte   0xa0
.byte    0x20
.uleb128 0xa
.4byte   $LASF25
.byte    0x5
.2byte   0x100
.4byte   0xa0
.byte    0x24
.uleb128 0xa
.4byte   $LASF26
.byte    0x5
.2byte   0x101
.4byte   0xa0
.byte    0x28
.uleb128 0xa
.4byte   $LASF27
.byte    0x5
.2byte   0x102

```

```

.4 byte  0xa0
. byte   0x2c
.uleb128 0xa
.4 byte  $LASF28
. byte   0x5
.2 byte  0x104
.4 byte  0x272
. byte   0x30
.uleb128 0xa
.4 byte  $LASF29
. byte   0x5
.2 byte  0x106
.4 byte  0x278
. byte   0x34
.uleb128 0xa
.4 byte  $LASF30
. byte   0x5
.2 byte  0x108
.4 byte  0x5a
. byte   0x38
.uleb128 0xa
.4 byte  $LASF31
. byte   0x5
.2 byte  0x10c
.4 byte  0x5a
. byte   0x3c
.uleb128 0xa
.4 byte  $LASF32
. byte   0x5
.2 byte  0x10e
.4 byte  0x7a
. byte   0x40
.uleb128 0xa
.4 byte  $LASF33
. byte   0x5
.2 byte  0x112
.4 byte  0x3e
. byte   0x44
.uleb128 0xa
.4 byte  $LASF34
. byte   0x5
.2 byte  0x113
.4 byte  0x4c
. byte   0x46
.uleb128 0xa
.4 byte  $LASF35
. byte   0x5
.2 byte  0x114
.4 byte  0x27e
. byte   0x47
.uleb128 0xa
.4 byte  $LASF36
. byte   0x5
.2 byte  0x118
.4 byte  0x28e
. byte   0x48

```



```
.uleb128 0xa
.4 byte $LASF37
.byte 0x5
.2 byte 0x121
.4 byte 0x8c
.byte 0x50
.uleb128 0xa
.4 byte $LASF38
.byte 0x5
.2 byte 0x129
.4 byte 0x9e
.byte 0x58
.uleb128 0xa
.4 byte $LASF39
.byte 0x5
.2 byte 0x12a
.4 byte 0x9e
.byte 0x5c
.uleb128 0xa
.4 byte $LASF40
.byte 0x5
.2 byte 0x12b
.4 byte 0x9e
.byte 0x60
.uleb128 0xa
.4 byte $LASF41
.byte 0x5
.2 byte 0x12c
.4 byte 0x9e
.byte 0x64
.uleb128 0xa
.4 byte $LASF42
.byte 0x5
.2 byte 0x12e
.4 byte 0x25
.byte 0x68
.uleb128 0xa
.4 byte $LASF43
.byte 0x5
.2 byte 0x12f
.4 byte 0x5a
.byte 0x6c
.uleb128 0xa
.4 byte $LASF44
.byte 0x5
.2 byte 0x131
.4 byte 0x294
.byte 0x70
.byte 0
.uleb128 0xb
.4 byte $LASF84
.byte 0x5
.byte 0x96
.uleb128 0x8
.4 byte $LASF46
.byte 0xc
```

```
. byte    0x5
. byte    0x9c
. 4 byte   0x272
. uleb128  0x9
. 4 byte   $LASF47
. byte    0x5
. byte    0x9d
. 4 byte   0x272
. byte    0
. uleb128  0x9
. 4 byte   $LASF48
. byte    0x5
. byte    0x9e
. 4 byte   0x278
. byte    0x4
. uleb128  0x9
. 4 byte   $LASF49
. byte    0x5
. byte    0xa2
. 4 byte   0x5a
. byte    0x8
. byte    0
. uleb128  0x6
. byte    0x4
. 4 byte   0x241
. uleb128  0x6
. byte    0x4
. 4 byte   0xbd
. uleb128  0xc
. 4 byte   0xa6
. 4 byte   0x28e
. uleb128  0xd
. 4 byte   0x97
. byte    0
. byte    0
. uleb128  0x6
. byte    0x4
. 4 byte   0x23a
. uleb128  0xc
. 4 byte   0xa6
. 4 byte   0x2a4
. uleb128  0xd
. 4 byte   0x97
. byte    0x27
. byte    0
. uleb128  0xe
. 4 byte   $LASF85
. uleb128  0xf
. 4 byte   $LASF50
. byte    0x5
. 2 byte   0x13b
. 4 byte   0x2a4
. uleb128  0xf
. 4 byte   $LASF51
. byte    0x5
. 2 byte   0x13c
```

```
.4 byte 0x2a4
.uleb128 0xf
.4 byte $LASF52
.byte 0x5
.2 byte 0x13d
.4 byte 0x2a4
.uleb128 0x6
.byte 0x4
.4 byte 0xad
.uleb128 0x7
.4 byte 0x2cd
.uleb128 0x10
.4 byte $LASF53
.byte 0x4
.byte 0xaa
.4 byte 0x278
.uleb128 0x10
.4 byte $LASF54
.byte 0x4
.byte 0xab
.4 byte 0x278
.uleb128 0x10
.4 byte $LASF55
.byte 0x4
.byte 0xac
.4 byte 0x278
.uleb128 0x10
.4 byte $LASF56
.byte 0x6
.byte 0x1a
.4 byte 0x5a
.uleb128 0xc
.4 byte 0x2d3
.4 byte 0x30f
.uleb128 0x11
.byte 0
.uleb128 0x7
.4 byte 0x304
.uleb128 0x10
.4 byte $LASF57
.byte 0x6
.byte 0x1b
.4 byte 0x30f
.uleb128 0x10
.4 byte $LASF58
.byte 0x7
.byte 0x39
.4 byte 0xa0
.uleb128 0x10
.4 byte $LASF59
.byte 0x7
.byte 0x47
.4 byte 0x5a
.uleb128 0x10
.4 byte $LASF60
.byte 0x7
```

```

.byte    0x4c
.4byte   0x5a
.uleb128 0x10
.4byte   $LASF61
.byte    0x7
.byte    0x50
.4byte   0x5a
.uleb128 0x8
.4byte   $LASF62
.byte    0x10
.byte    0x7
.byte    0x68
.4byte   0x388
.uleb128 0x9
.4byte   $LASF63
.byte    0x7
.byte    0x6a
.4byte   0x2cd
.byte    0
.uleb128 0x9
.4byte   $LASF64
.byte    0x7
.byte    0x6d
.4byte   0x5a
.byte    0x4
.uleb128 0x9
.4byte   $LASF65
.byte    0x7
.byte    0x6e
.4byte   0x388
.byte    0x8
.uleb128 0x12
.ascii   "val\000"
.byte    0x7
.byte    0x6f
.4byte   0x5a
.byte    0xc
.byte    0
.uleb128 0x6
.byte    0x4
.4byte   0x5a
.uleb128 0x3
.byte    0x8
.byte    0x4
.4byte   $LASF66
.uleb128 0x13
.4byte   $LASF86
.byte    0x1
.byte    0x68
.4byte   $LFB4
.4byte   $LFE4-$LFB4
.uleb128 0x1
.byte    0x9c
.4byte   0x3c8
.uleb128 0x14
.4byte   $LASF67

```

```

.byte    0x1
.byte    0x68
.4byte   0xa0
.uleb128 0x2
.byte    0x91
.sleb128 0
.uleb128 0x15
.4byte   $LASF68
.byte    0x1
.byte    0x69
.4byte   0x3c8
.uleb128 0x3
.byte    0x91
.sleb128 -160
.byte    0
.uleb128 0xc
.4byte   0xa6
.4byte   0x3d8
.uleb128 0xd
.4byte   0x97
.byte    0x95
.byte    0
.uleb128 0x16
.4byte   $LASF71
.byte    0x1
.byte    0x56
.4byte   0x5a
.4byte   $LFB3
.4byte   $LFE3-$LFB3
.uleb128 0x1
.byte    0x9c
.4byte   0x41c
.uleb128 0x14
.4byte   $LASF67
.byte    0x1
.byte    0x56
.4byte   0xa0
.uleb128 0x2
.byte    0x91
.sleb128 0
.uleb128 0x15
.4byte   $LASF69
.byte    0x1
.byte    0x57
.4byte   0x41c
.uleb128 0x2
.byte    0x91
.sleb128 -12
.uleb128 0x15
.4byte   $LASF70
.byte    0x1
.byte    0x5d
.4byte   0x5a
.uleb128 0x2
.byte    0x91
.sleb128 -16

```

```

.byte    0
.uleb128 0x6
.byte    0x4
.4byte   0xb2
.uleb128 0x16
.4byte   $LASF72
.byte    0x1
.byte    0xd
.4byte   0x5a
.4byte   $LFB2
.4byte   $LFE2-$LFB2
.uleb128 0x1
.byte    0x9c
.4byte   0x4bd
.uleb128 0x14
.4byte   $LASF73
.byte    0x1
.byte    0xd
.4byte   0x5a
.uleb128 0x2
.byte    0x91
.sleb128 0
.uleb128 0x14
.4byte   $LASF74
.byte    0x1
.byte    0xd
.4byte   0x4bd
.uleb128 0x2
.byte    0x91
.sleb128 4
.uleb128 0x15
.4byte   $LASF75
.byte    0x1
.byte    0xf
.4byte   0x41c
.uleb128 0x2
.byte    0x91
.sleb128 -32
.uleb128 0x15
.4byte   $LASF76
.byte    0x1
.byte    0x10
.4byte   0x41c
.uleb128 0x2
.byte    0x91
.sleb128 -28
.uleb128 0x15
.4byte   $LASF77
.byte    0x1
.byte    0x11
.4byte   0x5a
.uleb128 0x2
.byte    0x91
.sleb128 -24
.uleb128 0x15
.4byte   $LASF78

```

```

.byte    0x1
.byte    0x12
.4byte   0x5a
.uleb128 0x2
.byte    0x91
.sleb128 -16
.uleb128 0x17
.ascii   "opt\000"
.byte    0x1
.byte    0x13
.4byte   0x5a
.uleb128 0x2
.byte    0x91
.sleb128 -20
.uleb128 0x15
.4byte   $LASF79
.byte    0x1
.byte    0x14
.4byte   0x4c3
.uleb128 0x2
.byte    0x91
.sleb128 -12
.uleb128 0x15
.4byte   $LASF80
.byte    0x1
.byte    0x16
.4byte   0x4d3
.uleb128 0x5
.byte    0x3
.4byte   long_options.2593
.byte    0
.uleb128 0x6
.byte    0x4
.4byte   0xa0
.uleb128 0xc
.4byte   0xa6
.4byte   0x4d3
.uleb128 0xd
.4byte   0x97
.byte    0x1
.byte    0
.uleb128 0x18
.4byte   0x34b
.uleb128 0xd
.4byte   0x97
.byte    0x4
.byte    0
.byte    0
.section          .debug_abbrev,"",@progbits
$Ldebug_abbrev0:
.uleb128 0x1
.uleb128 0x11
.byte    0x1
.uleb128 0x25
.uleb128 0xe
.uleb128 0x13

```

```
.uleb128 0xb
.uleb128 0x3
.uleb128 0xe
.uleb128 0x1b
.uleb128 0xe
.uleb128 0x11
.uleb128 0x1
.uleb128 0x12
.uleb128 0x6
.uleb128 0x10
.uleb128 0x17
.byte 0
.byte 0
.uleb128 0x2
.uleb128 0x16
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x3
.uleb128 0x24
.byte 0
.uleb128 0xb
.uleb128 0xb
.uleb128 0x3e
.uleb128 0xb
.uleb128 0x3
.uleb128 0xe
.byte 0
.byte 0
.uleb128 0x4
.uleb128 0x24
.byte 0
.uleb128 0xb
.uleb128 0xb
.uleb128 0x3e
.uleb128 0xb
.uleb128 0x3
.uleb128 0x8
.byte 0
.byte 0
.uleb128 0x5
.uleb128 0xf
.byte 0
.uleb128 0xb
.uleb128 0xb
.byte 0
.byte 0
.uleb128 0x6
```



```
.uleb128 0xf
.byte 0
.uleb128 0xb
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x7
.uleb128 0x26
.byte 0
.uleb128 0x49
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x8
.uleb128 0x13
.byte 0x1
.uleb128 0x3
.uleb128 0xe
.uleb128 0xb
.uleb128 0xb
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x1
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x9
.uleb128 0xd
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.uleb128 0x38
.uleb128 0xb
.byte 0
.byte 0
.uleb128 0xa
.uleb128 0xd
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0x5
.uleb128 0x49
.uleb128 0x13
```

```
.uleb128 0x38
.uleb128 0xb
.byte 0
.byte 0
.uleb128 0xb
.uleb128 0x16
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.byte 0
.byte 0
.uleb128 0xc
.uleb128 0x1
.byte 0x1
.uleb128 0x49
.uleb128 0x13
.uleb128 0x1
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0xd
.uleb128 0x21
.byte 0
.uleb128 0x49
.uleb128 0x13
.uleb128 0x2f
.uleb128 0xb
.byte 0
.byte 0
.uleb128 0xe
.uleb128 0x13
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3c
.uleb128 0x19
.byte 0
.byte 0
.uleb128 0xf
.uleb128 0x34
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0x5
.uleb128 0x49
.uleb128 0x13
.uleb128 0x3f
.uleb128 0x19
.uleb128 0x3c
```

```
.uleb128 0x19
.byte 0
.byte 0
.uleb128 0x10
.uleb128 0x34
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.uleb128 0x3f
.uleb128 0x19
.uleb128 0x3c
.uleb128 0x19
.byte 0
.byte 0
.uleb128 0x11
.uleb128 0x21
.byte 0
.byte 0
.byte 0
.uleb128 0x12
.uleb128 0xd
.byte 0
.uleb128 0x3
.uleb128 0x8
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.uleb128 0x38
.uleb128 0xb
.byte 0
.byte 0
.uleb128 0x13
.uleb128 0x2e
.byte 0x1
.uleb128 0x3f
.uleb128 0x19
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x27
.uleb128 0x19
.uleb128 0x11
.uleb128 0x1
.uleb128 0x12
```

```
.uleb128 0x6
.uleb128 0x40
.uleb128 0x18
.uleb128 0x2116
.uleb128 0x19
.uleb128 0x1
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x14
.uleb128 0x5
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.uleb128 0x2
.uleb128 0x18
.byte 0
.byte 0
.uleb128 0x15
.uleb128 0x34
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.uleb128 0x2
.uleb128 0x18
.byte 0
.byte 0
.uleb128 0x16
.uleb128 0x2e
.byte 0x1
.uleb128 0x3f
.uleb128 0x19
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x27
.uleb128 0x19
.uleb128 0x49
.uleb128 0x13
.uleb128 0x11
.uleb128 0x1
```

```

.uleb128 0x12
.uleb128 0x6
.uleb128 0x40
.uleb128 0x18
.uleb128 0x2116
.uleb128 0x19
.uleb128 0x1
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x17
.uleb128 0x34
.byte 0
.uleb128 0x3
.uleb128 0x8
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.uleb128 0x2
.uleb128 0x18
.byte 0
.byte 0
.uleb128 0x18
.uleb128 0x1
.byte 0x1
.uleb128 0x49
.uleb128 0x13
.byte 0
.byte 0
.byte 0
.section .debug_aranges,"",@progbits
.4byte 0x1c
.2byte 0x2
.4byte $Ldebug_info0
.byte 0x4
.byte 0
.2byte 0
.2byte 0
.4byte $Ltext0
.4byte $Ltext0-$Ltext0
.4byte 0
.4byte 0
.section .debug_line,"",@progbits
$Ldebug_line0:
.section .debug_str,"MS",@progbits,1
$LASF10:
.ascii "__off_t\000"
$LASF17:
.ascii "_IO_read_ptr\000"
$LASF29:
.ascii "_chain\000"
$LASF8:
.ascii "size_t\000"

```

```

$LASF35:
    . ascii  "_shortbuf\000"
$LASF52:
    . ascii  "_IO_2_1_stderr_\000"
$LASF23:
    . ascii  "_IO_buf_base\000"
$LASF7:
    . ascii  "long_long_unsigned_int\000"
$LASF6:
    . ascii  "long_long_int\000"
$LASF4:
    . ascii  "signed_char\000"
$LASF77:
    . ascii  "flag_divisor\000"
$LASF86:
    . ascii  "notificar_problema_ruta\000"
$LASF30:
    . ascii  "_fileno\000"
$LASF18:
    . ascii  "_IO_read_end\000"
$LASF11:
    . ascii  "long_int\000"
$LASF80:
    . ascii  "long_options\000"
$LASF16:
    . ascii  "_flags\000"
$LASF24:
    . ascii  "_IO_buf_end\000"
$LASF33:
    . ascii  "_cur_column\000"
$LASF9:
    . ascii  "__quad_t\000"
$LASF79:
    . ascii  "estandar\000"
$LASF32:
    . ascii  "_old_offset\000"
$LASF37:
    . ascii  "_offset\000"
$LASF64:
    . ascii  "has_arg\000"
$LASF66:
    . ascii  "long_double\000"
$LASF46:
    . ascii  "_IO_marker\000"
$LASF53:
    . ascii  "stdin\000"
$LASF0:
    . ascii  "unsigned_int\000"
$LASF3:
    . ascii  "long_unsigned_int\000"
$LASF85:
    . ascii  "_IO_FILE_plus\000"
$LASF21:
    . ascii  "_IO_write_ptr\000"
$LASF63:
    . ascii  "name\000"

```

```

$LASF56:
    .ascii  "sys_nerr\000"
$LASF48:
    .ascii  "_sbuf\000"
$LASF2:
    .ascii  "short_unsigned_int\000"
$LASF25:
    .ascii  "_IO_save_base\000"
$LASF82:
    .ascii  "main.c\000"
$LASF36:
    .ascii  "_lock\000"
$LASF31:
    .ascii  "_flags2\000"
$LASF43:
    .ascii  "_mode\000"
$LASF54:
    .ascii  "stdout\000"
$LASF50:
    .ascii  "_IO_2_1_stdin_\000"
$LASF58:
    .ascii  "optarg\000"
$LASF81:
    .ascii  "GNU_C11_6.3.0_20170516_-meb_-march=mips32r2_-mfpxx_-mlls"
    .ascii  "c_-mno-lxc1-sxc1_-mips32r2_-mabi=32_-g\000"
$LASF13:
    .ascii  "sizetype\000"
$LASF83:
    .ascii  "/home/manu/TP1/OrgaDeCompus-TP1/Resolucion\000"
$LASF59:
    .ascii  "optind\000"
$LASF67:
    .ascii  "ruta\000"
$LASF22:
    .ascii  "_IO_write_end\000"
$LASF84:
    .ascii  "_IO_lock_t\000"
$LASF45:
    .ascii  "_IO_FILE\000"
$LASF70:
    .ascii  "character\000"
$LASF49:
    .ascii  "_pos\000"
$LASF68:
    .ascii  "mensaje\000"
$LASF57:
    .ascii  "sys_errlist\000"
$LASF78:
    .ascii  "long_index\000"
$LASF28:
    .ascii  "_markers\000"
$LASF1:
    .ascii  "unsigned_char\000"
$LASF5:
    .ascii  "short_int\000"
$LASF62:

```

```

        .ascii  "option\000"
$LASF65:
        .ascii  "flag\000"
$LASF34:
        .ascii  "_vtable_offset\000"
$LASF51:
        .ascii  "_IO_2_1_stdout_\000"
$LASF15:
        .ascii  "FILE\000"
$LASF61:
        .ascii  "optopt\000"
$LASF71:
        .ascii  "mostrar_en_pantalla\000"
$LASF14:
        .ascii  "char\000"
$LASF75:
        .ascii  "stream_entrada\000"
$LASF60:
        .ascii  "opterr\000"
$LASF47:
        .ascii  "_next\000"
$LASF12:
        .ascii  "__off64_t\000"
$LASF19:
        .ascii  "_IO_read_base\000"
$LASF27:
        .ascii  "_IO_save_end\000"
$LASF38:
        .ascii  "__pad1\000"
$LASF39:
        .ascii  "__pad2\000"
$LASF40:
        .ascii  "__pad3\000"
$LASF41:
        .ascii  "__pad4\000"
$LASF42:
        .ascii  "__pad5\000"
$LASF44:
        .ascii  "_unused2\000"
$LASF55:
        .ascii  "stderr\000"
$LASF74:
        .ascii  "argv\000"
$LASF69:
        .ascii  "archivo\000"
$LASF26:
        .ascii  "_IO_backup_base\000"
$LASF73:
        .ascii  "argc\000"
$LASF72:
        .ascii  "main\000"
$LASF20:
        .ascii  "_IO_write_base\000"
$LASF76:
        .ascii  "stream_salida\000"
        .ident  "GCC:_(Debian_6.3.0-18+deb9u1)_6.3.0_20170516"

```


7.2. max_divisor.s

```
.section .mdebug.abi32
.previous
.nan      legacy
.module fp=xx
.module nooddspreg
.abicalls
.text
$Ltext0:
.cfi_sections .debug_frame
.rdata
.align 2
$LC0:
.ascii "\012_Problema_asignando_memoria_dinamica.\012\000"
.align 2
$LC1:
.ascii "\012La_cantidad_de_numeros_ingresada_es_invalida._Solo_s"
.ascii "e_pueden_ingresar_2_numeros_por_cada_calculo_de_GCD.\012"
.ascii "\000"
.text
.align 2
.globl procesar_archivos
$LFB2 = .
.file 1 "max_divisor.c"
.loc 1 8 0
.cfi_startproc
.set      nomips16
.set      nomicromips
.ent      procesar_archivos
.type     procesar_archivos, @function
procesar_archivos:
.frame    $fp,64,$31          # vars= 32, regs= 2/0, args= 16, gp= 8
.mask     0xc0000000,-4
.fmask    0x00000000,0
.set      noreorder
.cpload   $25
.set      nomacro
addiu     $sp,$sp,-64
.cfi_def_cfa_offset 64
sw        $31,60($sp)
sw        $fp,56($sp)
.cfi_offset 31, -4
.cfi_offset 30, -8
move      $fp,$sp
.cfi_def_cfa_register 30
.cprestore 16
sw        $4,64($fp)
sw        $5,68($fp)
.loc 1 10 0
sw        $0,24($fp)
.loc 1 12 0
sw        $0,48($fp)
.loc 1 13 0
sw        $0,52($fp)
.loc 1 15 0
```

```

        li      $2,20                                # 0x14
        sw      $2,28($fp)
        .loc 1 16 0
        sw      $0,32($fp)
        .loc 1 17 0
        lw      $3,28($fp)
        move    $2,$3
        sll     $2,$2,1
        addu    $2,$2,$3
        sll     $2,$2,2
        move    $4,$2
        lw      $2,%call16(malloc)($28)
        move    $25,$2
        .reloc 1f,R_MIPS_JALR,malloc
1:      jalr    $25
        nop

$LVL0 = .
        lw      $28,16($fp)
        sw      $2,36($fp)
        .loc 1 18 0
        lw      $2,36($fp)
        bne     $2,$0,$L4
        nop

        .loc 1 19 0
        lw      $2,%got($LC0)($28)
        addiu   $4,$2,%lo($LC0)
        lw      $2,%call16(perror)($28)
        move    $25,$2
        .reloc 1f,R_MIPS_JALR,pererror
1:      jalr    $25
        nop

$LVL1 = .
        lw      $28,16($fp)
        .loc 1 20 0
        li      $2,-2                                # 0xfffffffffffffffe
        b       $L10
        nop

$L8:
        .loc 1 24 0
        lw      $2,28($fp)
        addiu   $2,$2,-1
        move    $3,$2
        lw      $2,32($fp)
        bne     $3,$2,$L5
        nop

        .loc 1 26 0
        lw      $2,28($fp)
        addiu   $2,$2,10
        sw      $2,28($fp)
        .loc 1 27 0
        lw      $3,28($fp)

```

```

        move    $2,$3
        sll     $2,$2,1
        addu    $2,$2,$3
        sll     $2,$2,2
        move    $5,$2
        lw      $4,36($fp)
        lw      $2,%call16(realloc)($28)
        move    $25,$2
        .reloc   1f,R_MIPS_JALR,realloc
1:      jalr    $25
        nop

$LVL2 = .
        lw      $28,16($fp)
        sw      $2,36($fp)
        .loc 1 29 0
        lw      $2,36($fp)
        bne     $2,$0,$L5
        nop

        .loc 1 30 0
        lw      $2,%got($LC0)($28)
        addiu   $4,$2,%lo($LC0)
        lw      $2,%call16(perror)($28)
        move    $25,$2
        .reloc   1f,R_MIPS_JALR,pererr
1:      jalr    $25
        nop

$LVL3 = .
        lw      $28,16($fp)
        .loc 1 31 0
        li      $2,-2                      # 0xfffffffffffffe
        b       $L10
        nop

$L5:
        .loc 1 35 0
        addiu   $3,$fp,52
        addiu   $2,$fp,48
        move    $6,$3
        move    $5,$2
        lw      $4,64($fp)
        lw      $2,%got(leer_linea)($28)
        move    $25,$2
        .reloc   1f,R_MIPS_JALR,leer_linea
1:      jalr    $25
        nop

$LVL4 = .
        lw      $28,16($fp)
        sw      $2,24($fp)
        .loc 1 37 0
        lw      $3,24($fp)
        li      $2,-3                      # 0xfffffffffffffd
        bne     $3,$2,$L6

```

```

        nop

        .loc 1 38 0
        lw      $2,%got($LC1)($28)
        addiu   $4,$2,%lo($LC1)
        lw      $2,%call16(perror)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,pererror
1:        jalr   $25
        nop

$LVL5 = .
        lw      $28,16($fp)
        .loc 1 39 0
        li      $2,1                # 0x1
        b       $L10
        nop

$L6:
        .loc 1 42 0
        lw      $3,24($fp)
        li      $2,-1              # 0xffffffffffffffff
        beq     $3,$2,$L4
        nop

        .loc 1 44 0
        lw      $2,52($fp)
        lw      $3,48($fp)
        addiu   $4,$fp,40
        move    $6,$4
        move    $5,$3
        move    $4,$2
        lw      $2,%got(pasar_a_enteros)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,pasar_a_enteros
1:        jalr   $25
        nop

$LVL6 = .
        lw      $28,16($fp)
        beq     $2,$0,$L7
        nop

        .loc 1 45 0
        li      $2,1                # 0x1
        b       $L10
        nop

$L7:
        .loc 1 52 0
        lw      $3,32($fp)
        move    $2,$3
        sll     $2,$2,1
        addu    $2,$2,$3
        sll     $2,$2,2
        move    $3,$2

```

```

        lw      $2,36($fp)
        addu    $2,$2,$3
        lw      $3,40($fp)
        sw      $3,0($2)
        .loc 1 53 0
        lw      $3,32($fp)
        move    $2,$3
        sll     $2,$2,1
        addu    $2,$2,$3
        sll     $2,$2,2
        move    $3,$2
        lw      $2,36($fp)
        addu    $2,$2,$3
        lw      $3,44($fp)
        sw      $3,4($2)
        .loc 1 54 0
        lw      $3,32($fp)
        move    $2,$3
        sll     $2,$2,1
        addu    $2,$2,$3
        sll     $2,$2,2
        move    $3,$2
        lw      $2,36($fp)
        addu    $2,$2,$3
        sw      $0,8($2)
        .loc 1 55 0
        lw      $2,32($fp)
        addiu   $2,$2,1
        sw      $2,32($fp)
$L4:
        .loc 1 23 0
        lw      $2,24($fp)
        beq     $2,$0,$L8
        nop

        .loc 1 59 0
        lw      $3,24($fp)
        li      $2,-1                # 0xffffffffffffffff
        bne     $3,$2,$L9
        nop

        .loc 1 59 0 is_stmt 0 discriminator 1
        li      $2,1                # 0x1
        b       $L10
        nop

$L9:
        .loc 1 61 0 is_stmt 1
        lw      $5,32($fp)
        lw      $4,36($fp)
        lw      $2,%call16(euclides)($28)
        move    $25,$2
        .reloc   1f,R_MIPS_JALR,euclides
1:         jalr  $25
        nop

```

```

$LVL7 = .
    lw      $28,16($fp)
    .loc 1 62 0
    lw      $6,68($fp)
    lw      $5,32($fp)
    lw      $4,36($fp)
    lw      $2,%got(imprimir_salida)($28)
    move     $25,$2
    .reloc   1f,R_MIPS_JALR,imprimir_salida
1:    jalr   $25
    nop

$LVL8 = .
    lw      $28,16($fp)
    .loc 1 63 0
    lw      $4,36($fp)
    lw      $2,%call16(free)($28)
    move     $25,$2
    .reloc   1f,R_MIPS_JALR,free
1:    jalr   $25
    nop

$LVL9 = .
    lw      $28,16($fp)
    .loc 1 64 0
    lw      $2,52($fp)
    move     $4,$2
    lw      $2,%call16(free)($28)
    move     $25,$2
    .reloc   1f,R_MIPS_JALR,free
1:    jalr   $25
    nop

$LVL10 = .
    lw      $28,16($fp)
    .loc 1 65 0
    move     $2,$0
$L10:
    .loc 1 66 0 discriminator 1
    move     $sp,$fp
    .cfi_def_cfa_register 29
    lw      $31,60($sp)
    lw      $fp,56($sp)
    addiu    $sp,$sp,64
    .cfi_restore 30
    .cfi_restore 31
    .cfi_def_cfa_offset 0
    jr      $31
    nop

    .set     macro
    .set     reorder
    .end     procesar_archivos
    .cfi_endproc

$LFE2:
    .size    procesar_archivos , .-procesar_archivos

```

```

        .rdata
        .align 2
$LC2:
        .ascii  "\012Problema_leyendo_el_archivo.\012\000"
        .text
        .align 2
        .globl leer_linea
$LFB3 = .
        .loc 1 68 0
        .cfi_startproc
        .set      nomips16
        .set      nomicromips
        .ent      leer_linea
        .type     leer_linea , @function
leer_linea:
        .frame    $fp,40,$31           # vars= 8, regs= 2/0, args= 16, gp= 8
        .mask     0xc0000000,-4
        .fmask    0x00000000,0
        .set      noreorder
        .cpload   $25
        .set      nomacro
        addiu     $sp,$sp,-40
        .cfi_def_cfa_offset 40
        sw        $31,36($sp)
        sw        $fp,32($sp)
        .cfi_offset 31, -4
        .cfi_offset 30, -8
        move      $fp,$sp
        .cfi_def_cfa_register 30
        .cprestore 16
        sw        $4,40($fp)
        sw        $5,44($fp)
        sw        $6,48($fp)
        .loc 1 69 0
        li        $2,20                # 0x14
        sw        $2,24($fp)
        .loc 1 70 0
        lw        $2,24($fp)
        move      $4,$2
        lw        $2,%call16(malloc)($28)
        move      $25,$2
        .reloc    1f,R_MIPS_JALR,malloc
1:        jalr     $25
        nop

$LVL11 = .
        lw        $28,16($fp)
        move      $3,$2
        lw        $2,48($fp)
        sw        $3,0($2)
        .loc 1 72 0
        lw        $2,48($fp)
        lw        $2,0($2)
        bne       $2,$0,$L12
        nop

```

```

        .loc 1 73 0
        li      $2,-2                # 0xfffffffffffffffe
        b       $L13
        nop

$L12:
        .loc 1 76 0
        lw      $2,44($fp)
        sw      $0,0($2)
        .loc 1 77 0
        li      $2,1                 # 0x1
        sw      $2,28($fp)
        .loc 1 79 0
        b       $L14
        nop

$L19:
        .loc 1 81 0
        lw      $2,44($fp)
        lw      $3,0($2)
        lw      $2,24($fp)
        addiu   $2,$2,-1
        bne     $3,$2,$L15
        nop

        .loc 1 82 0
        lw      $2,24($fp)
        addiu   $2,$2,10
        sw      $2,24($fp)
        .loc 1 83 0
        lw      $2,48($fp)
        lw      $2,0($2)
        lw      $3,24($fp)
        move    $5,$3
        move    $4,$2
        lw      $2,%call16(realloc)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,realloc
1:      jalr    $25
        nop

$LVL12 = .
        lw      $28,16($fp)
        move    $3,$2
        lw      $2,48($fp)
        sw      $3,0($2)
        .loc 1 84 0
        lw      $2,48($fp)
        lw      $2,0($2)
        bne     $2,$0,$L15
        nop

        .loc 1 85 0
        li      $2,-2                # 0xfffffffffffffffe
        b       $L13
        nop

```



```

$L15:
    .loc 1 89 0
    lw      $4,40($fp)
    lw      $2,%call16(ferror)($28)
    move    $25,$2
    .reloc 1f,R_MIPS_JALR,ferror
1:    jalr   $25
    nop

$LVL13 = .
    lw      $28,16($fp)
    beq     $2,$0,$L16
    nop

    .loc 1 90 0
    lw      $2,%got($LC2)($28)
    addiu   $4,$2,%lo($LC2)
    lw      $2,%call16(perror)($28)
    move    $25,$2
    .reloc 1f,R_MIPS_JALR,pererror
1:    jalr   $25
    nop

$LVL14 = .
    lw      $28,16($fp)
    .loc 1 91 0
    li      $2,-1                # 0xffffffffffffffff
    b       $L13
    nop

$L16:
    .loc 1 94 0
    lw      $4,40($fp)
    lw      $2,%call16(_IO_getc)($28)
    move    $25,$2
    .reloc 1f,R_MIPS_JALR,_IO_getc
1:    jalr   $25
    nop

$LVL15 = .
    lw      $28,16($fp)
    sw      $2,28($fp)
    .loc 1 95 0
    lw      $2,28($fp)
    seb     $2,$2
    move    $4,$2
    lw      $2,%got(es_caracter_invalido)($28)
    move    $25,$2
    .reloc 1f,R_MIPS_JALR,es_caracter_invalido
1:    jalr   $25
    nop

$LVL16 = .
    lw      $28,16($fp)
    beq     $2,$0,$L17

```

```

nop

.loc 1 96 0
li    $2,-1                # 0xffffffffffffffff
b     $L13
nop

$L17:
.loc 1 98 0
lw     $2,48($fp)
lw     $2,0($2)
lw     $3,44($fp)
lw     $3,0($3)
addu   $2,$2,$3
lw     $3,28($fp)
seb    $3,$3
sb     $3,0($2)
.loc 1 99 0
lw     $2,44($fp)
lw     $2,0($2)
addiu  $3,$2,1
lw     $2,44($fp)
sw     $3,0($2)

$L14:
.loc 1 79 0
lw     $3,28($fp)
li     $2,10                # 0xa
beq    $3,$2,$L18
nop

.loc 1 79 0 is_stmt 0 discriminator 1
lw     $3,28($fp)
li     $2,-1                # 0xffffffffffffffff
bne    $3,$2,$L19
nop

$L18:
.loc 1 102 0 is_stmt 1
lw     $3,28($fp)
li     $2,-1                # 0xffffffffffffffff
beq    $3,$2,$L20
nop

.loc 1 102 0 is_stmt 0 discriminator 1
lw     $2,44($fp)
lw     $2,0($2)
slt    $2,$2,2
beq    $2,$0,$L21
nop

$L20:
.loc 1 103 0 is_stmt 1
li     $2,1                 # 0x1
b     $L13
nop

```

```

$L21:
    .loc 1 106 0
    move    $2,$0
$L13:
    .loc 1 108 0
    move    $sp,$fp
    .cfi_def_cfa_register 29
    lw      $31,36($sp)
    lw      $fp,32($sp)
    addiu   $sp,$sp,40
    .cfi_restore 30
    .cfi_restore 31
    .cfi_def_cfa_offset 0
    jr      $31
    nop

    .set     macro
    .set     reorder
    .end     leer_linea
    .cfi_endproc

$LFE3:
    .size    leer_linea , .-leer_linea
    .align   2
    .globl   pasar_a_enteros
$LFB4 = .
    .loc 1 110 0
    .cfi_startproc
    .set     nomips16
    .set     nomicromips
    .ent     pasar_a_enteros
    .type    pasar_a_enteros , @function
pasar_a_enteros:
    .frame   $fp,64,$31                # vars= 24, regs= 4/0, args= 16, gp= 8
    .mask    0xc0030000,-4
    .fmask   0x00000000,0
    .set     noreorder
    .cpload  $25
    .set     nomacro
    addiu    $sp,$sp,-64
    .cfi_def_cfa_offset 64
    sw       $31,60($sp)
    sw       $fp,56($sp)
    sw       $17,52($sp)
    sw       $16,48($sp)
    .cfi_offset 31, -4
    .cfi_offset 30, -8
    .cfi_offset 17, -12
    .cfi_offset 16, -16
    move     $fp,$sp
    .cfi_def_cfa_register 30
    .cprestore 16
    sw       $4,64($fp)
    sw       $5,68($fp)
    sw       $6,72($fp)
    .loc 1 110 0
    move     $4,$sp

```

```

move    $17,$4
.loc 1 111 0
lw      $4,68($fp)
addiu   $5,$4,-1
sw      $5,36($fp)
move    $5,$4
move    $13,$5
move    $12,$0
srl     $5,$13,29
sll     $8,$12,3
or      $8,$5,$8
sll     $9,$13,3
move    $5,$4
move    $11,$5
move    $10,$0
srl     $5,$11,29
sll     $2,$10,3
or      $2,$5,$2
sll     $3,$11,3
move    $2,$4
addiu   $2,$2,7
srl     $2,$2,3
sll     $2,$2,3
subu    $sp,$sp,$2
addiu   $2,$sp,16
addiu   $2,$2,0
sw      $2,40($fp)
.loc 1 112 0
li      $2,65
sb      $2,44($fp)
.loc 1 113 0
sw      $0,24($fp)
.loc 1 115 0
sw      $0,28($fp)
.loc 1 116 0
sw      $0,32($fp)
.loc 1 118 0
b       $L23
nop

```

0x41

\$L27:

```

.loc 1 119 0
lw      $2,28($fp)
lw      $3,64($fp)
addu    $2,$3,$2
lbu     $2,0($2)
sb      $2,44($fp)
lw      $2,28($fp)
addiu   $2,$2,1
sw      $2,28($fp)
.loc 1 120 0
lw      $2,24($fp)
slt     $2,$2,2
bne     $2,$0,$L24
nop

```

```

        .loc 1 121 0
        lw      $2,%got($LC1)($28)
        addiu   $4,$2,%lo($LC1)
        lw      $2,%call16(perror)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,pererror
1:      jalr    $25
        nop

$LV17 = .
        lw      $28,16($fp)
        .loc 1 122 0
        li      $2,-3                # 0xffffffffffffd
        b       $L25
        nop

$L24:
        .loc 1 123 0
        lb      $2,44($fp)
        move    $4,$2
        lw      $2,%got(es_numerico)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,es_numerico
1:      jalr    $25
        nop

$LV18 = .
        lw      $28,16($fp)
        beq     $2,$0,$L26
        nop

        .loc 1 124 0
        lw      $3,40($fp)
        lw      $2,32($fp)
        addu    $2,$3,$2
        lbu     $3,44($fp)
        sb      $3,0($2)
        lw      $2,32($fp)
        addiu   $2,$2,1
        sw      $2,32($fp)
        b       $L23
        nop

$L26:
        .loc 1 125 0
        li      $2,1                # 0x1
        sb      $2,44($fp)
        lw      $2,32($fp)
        beq     $2,$0,$L23
        nop

        .loc 1 126 0
        lw      $3,40($fp)
        lw      $2,32($fp)
        addu    $2,$3,$2
        sb      $0,0($2)

```

```

        .loc 1 127 0
        lw      $2,24($fp)
        sll     $2,$2,2
        lw      $3,72($fp)
        addu    $16,$3,$2
        lw      $2,40($fp)
        move    $4,$2
        lw      $2,%call16(atoi)($28)
        move    $25,$2
        .reloc   1f,R_MIPS_JALR,atoi
1:      jalr    $25
        nop

$LV19 = .
        lw      $28,16($fp)
        sw      $2,0($16)
        .loc 1 128 0
        lw      $2,24($fp)
        addiu   $2,$2,1
        sw      $2,24($fp)
        .loc 1 129 0
        sw      $0,32($fp)

$L23:
        .loc 1 118 0
        lw      $3,28($fp)
        lw      $2,68($fp)
        slt     $2,$3,$2
        bne     $2,$0,$L27
        nop

        .loc 1 132 0
        move    $2,$0

$L25:
        move    $sp,$17
        .loc 1 133 0
        move    $sp,$fp
        .cfi_def_cfa_register 29
        lw      $31,60($sp)
        lw      $fp,56($sp)
        lw      $17,52($sp)
        lw      $16,48($sp)
        addiu   $sp,$sp,64
        .cfi_restore 16
        .cfi_restore 17
        .cfi_restore 30
        .cfi_restore 31
        .cfi_def_cfa_offset 0
        jr      $31
        nop

        .set     macro
        .set     reorder
        .end     pasar_a_enteros
        .cfi_endproc

$LFE4:
        .size    pasar_a_enteros , .-pasar_a_enteros

```

```

        .rdata
        .align 2
$LC3:
        .ascii  "GCD(%i,%i)=%i\012\000"
        .text
        .align 2
        .globl  imprimir_salida
$LFB5 = .
        .loc 1 135 0
        .cfi_startproc
        .set     nomips16
        .set     nomicromips
        .ent     imprimir_salida
        .type    imprimir_salida, @function
imprimir_salida:
        .frame   $fp,48,$31           # vars= 8, regs= 2/0, args= 24, gp= 8
        .mask    0xc0000000,-4
        .fmask    0x00000000,0
        .set     noreorder
        .cpload   $25
        .set     nomacro
        addiu    $sp,$sp,-48
        .cfi_def_cfa_offset 48
        sw       $31,44($sp)
        sw       $fp,40($sp)
        .cfi_offset 31,-4
        .cfi_offset 30,-8
        move     $fp,$sp
        .cfi_def_cfa_register 30
        .cprestore 24
        sw       $4,48($fp)
        sw       $5,52($fp)
        sw       $6,56($fp)
        .loc 1 136 0
        lw       $2,52($fp)
        beq      $2,$0,$L34
        nop

$LBB2 = .
        .loc 1 138 0
        sw       $0,32($fp)
        b        $L32
        nop

$L33:
        .loc 1 139 0 discriminator 3
        lw       $3,32($fp)
        move     $2,$3
        sll      $2,$2,1
        addu     $2,$2,$3
        sll      $2,$2,2
        move     $3,$2
        lw       $2,48($fp)
        addu     $2,$2,$3
        lw       $4,0($2)
        lw       $3,32($fp)

```

```

        move    $2,$3
        sll     $2,$2,1
        addu    $2,$2,$3
        sll     $2,$2,2
        move    $3,$2
        lw      $2,48($fp)
        addu    $2,$2,$3
        lw      $5,4($2)
        lw      $3,32($fp)
        move    $2,$3
        sll     $2,$2,1
        addu    $2,$2,$3
        sll     $2,$2,2
        move    $3,$2
        lw      $2,48($fp)
        addu    $2,$2,$3
        lw      $2,8($2)
        sw      $2,16($sp)
        move    $7,$5
        move    $6,$4
        lw      $2,%got($LC3)($28)
        addiu   $5,$2,%lo($LC3)
        lw      $4,56($fp)
        lw      $2,%call16(fprintf)($28)
        move    $25,$2
        .reloc   1f,R_MIPS_JALR,fprintf
1:      jalr    $25
        nop

$VL20 = .
        lw      $28,24($fp)
        .loc 1 138 0 discriminator 3
        lw      $2,32($fp)
        addiu   $2,$2,1
        sw      $2,32($fp)

$L32:
        .loc 1 138 0 is_stmt 0 discriminator 1
        lw      $2,52($fp)
        addiu   $3,$2,-1
        lw      $2,32($fp)
        sltu    $2,$2,$3
        bne     $2,$0,$L33
        nop

        b       $L29
        nop

$L34:
$LBE2 = .
        .loc 1 136 0 is_stmt 1
        nop

$L29:
        .loc 1 141 0
        move    $sp,$fp
        .cfi_def_cfa_register 29
        lw      $31,44($sp)

```



```

        lw      $fp,40($sp)
        addiu   $sp,$sp,48
        .cfi_restore 30
        .cfi_restore 31
        .cfi_def_cfa_offset 0
        jr      $31
        nop

        .set     macro
        .set     reorder
        .end      imprimir_salida
        .cfi_endproc

$LFE5:
        .size    imprimir_salida, .-imprimir_salida
        .align   2
        .globl   es_fin_de_linea
$LFB6 = .
        .loc 1 143 0
        .cfi_startproc
        .set     nomips16
        .set     nomicromips
        .ent      es_fin_de_linea
        .type     es_fin_de_linea, @function
es_fin_de_linea:
        .frame    $fp,8,$31                # vars= 0, regs= 1/0, args= 0, gp= 0
        .mask     0x40000000,-4
        .fmask    0x00000000,0
        .set      noreorder
        .set      nomacro
        addiu     $sp,$sp,-8
        .cfi_def_cfa_offset 8
        sw        $fp,4($sp)
        .cfi_offset 30, -4
        move      $fp,$sp
        .cfi_def_cfa_register 30
        move      $2,$4
        sb        $2,8($fp)
        .loc 1 144 0
        lb        $3,8($fp)
        li        $2,10                    # 0xa
        beq       $3,$2,$L36
        nop

        .loc 1 144 0 is_stmt 0 discriminator 2
        lb        $3,8($fp)
        li        $2,-1                    # 0xffffffffffffffff
        bne       $3,$2,$L37
        nop

$L36:
        .loc 1 144 0 discriminator 3
        li        $2,1                    # 0x1
        b         $L38
        nop

$L37:

```

```

        .loc 1 144 0 discriminator 4
        move    $2,$0
$L38:
        .loc 1 144 0 discriminator 6
        andi    $2,$2,0x1
        andi    $2,$2,0x00ff
        .loc 1 145 0 is_stmt 1 discriminator 6
        move    $sp,$fp
        .cfi_def_cfa_register 29
        lw      $fp,4($sp)
        addiu   $sp,$sp,8
        .cfi_restore 30
        .cfi_def_cfa_offset 0
        jr      $31
        nop

        .set     macro
        .set     reorder
        .end     es_fin_de_linea
        .cfi_endproc

$LFE6:
        .size    es_fin_de_linea , .-es_fin_de_linea
        .align   2
        .globl   es_numerico
$LFB7 = .
        .loc 1 147 0
        .cfi_startproc
        .set     nomips16
        .set     nomicromips
        .ent     es_numerico
        .type    es_numerico , @function
es_numerico:
        .frame   $fp,8,$31                # vars= 0, regs= 1/0, args= 0, gp= 0
        .mask    0x40000000,-4
        .fmask    0x00000000,0
        .set     noreorder
        .set     nomacro
        addiu    $sp,$sp,-8
        .cfi_def_cfa_offset 8
        sw       $fp,4($sp)
        .cfi_offset 30, -4
        move     $fp,$sp
        .cfi_def_cfa_register 30
        move     $2,$4
        sb       $2,8($fp)
        .loc 1 148 0
        lb       $2,8($fp)
        slt      $2,$2,48
        bne      $2,$0,$L41
        nop

        .loc 1 148 0 is_stmt 0 discriminator 1
        lb       $2,8($fp)
        slt      $2,$2,58
        bne      $2,$0,$L42
        nop

```

```

$L41:
    .loc 1 148 0 discriminator 4
    lb     $3,8($fp)
    li     $2,45                # 0x2d
    beq    $3,$2,$L42
    nop

    .loc 1 148 0 discriminator 6
    lb     $3,8($fp)
    li     $2,43                # 0x2b
    bne    $3,$2,$L43
    nop

$L42:
    .loc 1 148 0 discriminator 7
    li     $2,1                # 0x1
    b      $L44
    nop

$L43:
    .loc 1 148 0 discriminator 8
    move    $2,$0

$L44:
    .loc 1 148 0 discriminator 10
    andi    $2,$2,0x1
    andi    $2,$2,0x00ff
    .loc 1 149 0 is_stmt 1 discriminator 10
    move    $sp,$fp
    .cfi_def_cfa_register 29
    lw      $fp,4($sp)
    addiu   $sp,$sp,8
    .cfi_restore 30
    .cfi_def_cfa_offset 0
    jr      $31
    nop

    .set    macro
    .set    reorder
    .end    es_numerico
    .cfi_endproc

$LFE7:
    .size   es_numerico, .-es_numerico
    .align  2
    .globl  es_caracter_invalido

$LFB8 = .
    .loc 1 151 0
    .cfi_startproc
    .set    nomips16
    .set    nomicromips
    .ent    es_caracter_invalido
    .type   es_caracter_invalido, @function
es_caracter_invalido:
    .frame  $fp,32,$31          # vars= 0, regs= 2/0, args= 16, gp= 8
    .mask   0xc0000000,-4
    .fmask  0x00000000,0

```

```

.set      noreorder
.cpload  $25
.set      nomacro
addiu     $sp,$sp,-32
.cfi_def_cfa_offset 32
sw        $31,28($sp)
sw        $fp,24($sp)
.cfi_offset 31,-4
.cfi_offset 30,-8
move      $fp,$sp
.cfi_def_cfa_register 30
.cprestore 16
move      $2,$4
sb        $2,32($fp)
.loc 1 152 0
lb        $2,32($fp)
move      $4,$2
lw        $2,%got(es_numerico)($28)
move      $25,$2
.reloc    1f,R_MIPS_JALR,es_numerico
1:        jalr    $25
        nop

$LVL21 = .
        lw        $28,16($fp)
        xori      $2,$2,0x1
        andi      $2,$2,0x00ff
        beq       $2,$0,$L47
        nop

        .loc 1 152 0 is_stmt 0 discriminator 1
        lb        $2,32($fp)
        move      $4,$2
        lw        $2,%got(es_fin_de_linea)($28)
        move      $25,$2
        .reloc    1f,R_MIPS_JALR,es_fin_de_linea
1:        jalr    $25
        nop

$LVL22 = .
        lw        $28,16($fp)
        xori      $2,$2,0x1
        andi      $2,$2,0x00ff
        beq       $2,$0,$L47
        nop

        .loc 1 152 0 discriminator 3
        lb        $3,32($fp)
        li        $2,32                # 0x20
        beq       $3,$2,$L47
        nop

        .loc 1 152 0 discriminator 5
        li        $2,1                # 0x1
        b         $L48
        nop

```

```

$L47:
    .loc 1 152 0 discriminator 6
    move    $2,$0
$L48:
    .loc 1 152 0 discriminator 8
    andi    $2,$2,0x1
    andi    $2,$2,0x00ff
    .loc 1 153 0 is_stmt 1 discriminator 8
    move    $sp,$fp
    .cfi_def_cfa_register 29
    lw      $31,28($sp)
    lw      $fp,24($sp)
    addiu   $sp,$sp,32
    .cfi_restore 30
    .cfi_restore 31
    .cfi_def_cfa_offset 0
    jr      $31
    nop

    .set    macro
    .set    reorder
    .end    es_caracter_invalido
    .cfi_endproc
$LFE8:
    .size    es_caracter_invalido , .-es_caracter_invalido
$Ltext0:
    .file 2 "/usr/lib/gcc/mips-linux-gnu/6/include/stddef.h"
    .file 3 "/usr/include/mips-linux-gnu/bits/types.h"
    .file 4 "/usr/include/stdio.h"
    .file 5 "/usr/include/libio.h"
    .file 6 "/usr/include/mips-linux-gnu/bits/sys_errlist.h"
    .file 7 "/usr/include/getopt.h"
    .file 8 "gcd.h"
    .section          .debug_info,"",@progbits
$Ldebug_info0:
    .4byte 0x60b
    .2byte 0x4
    .4byte $Ldebug_abbrev0
    .byte 0x4
    .uleb128 0x1
    .4byte $LASF88
    .byte 0xc
    .4byte $LASF89
    .4byte $LASF90
    .4byte $Ltext0
    .4byte $Ltext0-$Ltext0
    .4byte $Ldebug_line0
    .uleb128 0x2
    .4byte $LASF8
    .byte 0x2
    .byte 0xd8
    .4byte 0x30
    .uleb128 0x3
    .byte 0x4
    .byte 0x7

```

```

.4byte $LASF0
.uleb128 0x3
.byte 0x1
.byte 0x8
.4byte $LASF1
.uleb128 0x3
.byte 0x2
.byte 0x7
.4byte $LASF2
.uleb128 0x3
.byte 0x4
.byte 0x7
.4byte $LASF3
.uleb128 0x3
.byte 0x1
.byte 0x6
.4byte $LASF4
.uleb128 0x3
.byte 0x2
.byte 0x5
.4byte $LASF5
.uleb128 0x4
.byte 0x4
.byte 0x5
.ascii "int\000"
.uleb128 0x3
.byte 0x8
.byte 0x5
.4byte $LASF6
.uleb128 0x3
.byte 0x8
.byte 0x7
.4byte $LASF7
.uleb128 0x2
.4byte $LASF9
.byte 0x3
.byte 0x37
.4byte 0x61
.uleb128 0x2
.4byte $LASF10
.byte 0x3
.byte 0x83
.4byte 0x85
.uleb128 0x3
.byte 0x4
.byte 0x5
.4byte $LASF11
.uleb128 0x2
.4byte $LASF12
.byte 0x3
.byte 0x84
.4byte 0x6f
.uleb128 0x5
.4byte 0x5a
.4byte 0xa7
.uleb128 0x6

```

```
.4byte 0xa7
.byte 0x1
.byte 0
.uleb128 0x3
.byte 0x4
.byte 0x7
.4byte $LASF13
.uleb128 0x7
.byte 0x4
.uleb128 0x8
.byte 0x4
.4byte 0xb6
.uleb128 0x3
.byte 0x1
.byte 0x6
.4byte $LASF14
.uleb128 0x9
.4byte 0xb6
.uleb128 0x2
.4byte $LASF15
.byte 0x4
.byte 0x30
.4byte 0xcd
.uleb128 0xa
.4byte $LASF45
.byte 0x98
.byte 0x5
.byte 0xf1
.4byte 0x24a
.uleb128 0xb
.4byte $LASF16
.byte 0x5
.byte 0xf2
.4byte 0x5a
.byte 0
.uleb128 0xb
.4byte $LASF17
.byte 0x5
.byte 0xf7
.4byte 0xb0
.byte 0x4
.uleb128 0xb
.4byte $LASF18
.byte 0x5
.byte 0xf8
.4byte 0xb0
.byte 0x8
.uleb128 0xb
.4byte $LASF19
.byte 0x5
.byte 0xf9
.4byte 0xb0
.byte 0xc
.uleb128 0xb
.4byte $LASF20
.byte 0x5
```

```

.byte    0xfa
.4byte   0xb0
.byte    0x10
.uleb128 0xb
.4byte   $LASF21
.byte    0x5
.byte    0xfb
.4byte   0xb0
.byte    0x14
.uleb128 0xb
.4byte   $LASF22
.byte    0x5
.byte    0xfc
.4byte   0xb0
.byte    0x18
.uleb128 0xb
.4byte   $LASF23
.byte    0x5
.byte    0xfd
.4byte   0xb0
.byte    0x1c
.uleb128 0xb
.4byte   $LASF24
.byte    0x5
.byte    0xfe
.4byte   0xb0
.byte    0x20
.uleb128 0xc
.4byte   $LASF25
.byte    0x5
.2byte   0x100
.4byte   0xb0
.byte    0x24
.uleb128 0xc
.4byte   $LASF26
.byte    0x5
.2byte   0x101
.4byte   0xb0
.byte    0x28
.uleb128 0xc
.4byte   $LASF27
.byte    0x5
.2byte   0x102
.4byte   0xb0
.byte    0x2c
.uleb128 0xc
.4byte   $LASF28
.byte    0x5
.2byte   0x104
.4byte   0x282
.byte    0x30
.uleb128 0xc
.4byte   $LASF29
.byte    0x5
.2byte   0x106
.4byte   0x288

```



```
. byte    0x34
. uleb128 0xc
. 4 byte   $LASF30
. byte    0x5
. 2 byte   0x108
. 4 byte   0x5a
. byte    0x38
. uleb128 0xc
. 4 byte   $LASF31
. byte    0x5
. 2 byte   0x10c
. 4 byte   0x5a
. byte    0x3c
. uleb128 0xc
. 4 byte   $LASF32
. byte    0x5
. 2 byte   0x10e
. 4 byte   0x7a
. byte    0x40
. uleb128 0xc
. 4 byte   $LASF33
. byte    0x5
. 2 byte   0x112
. 4 byte   0x3e
. byte    0x44
. uleb128 0xc
. 4 byte   $LASF34
. byte    0x5
. 2 byte   0x113
. 4 byte   0x4c
. byte    0x46
. uleb128 0xc
. 4 byte   $LASF35
. byte    0x5
. 2 byte   0x114
. 4 byte   0x28e
. byte    0x47
. uleb128 0xc
. 4 byte   $LASF36
. byte    0x5
. 2 byte   0x118
. 4 byte   0x29e
. byte    0x48
. uleb128 0xc
. 4 byte   $LASF37
. byte    0x5
. 2 byte   0x121
. 4 byte   0x8c
. byte    0x50
. uleb128 0xc
. 4 byte   $LASF38
. byte    0x5
. 2 byte   0x129
. 4 byte   0xae
. byte    0x58
. uleb128 0xc
```

```

.4 byte  $LASF39
. byte   0x5
.2 byte  0x12a
.4 byte  0xae
. byte   0x5c
.uleb128 0xc
.4 byte  $LASF40
. byte   0x5
.2 byte  0x12b
.4 byte  0xae
. byte   0x60
.uleb128 0xc
.4 byte  $LASF41
. byte   0x5
.2 byte  0x12c
.4 byte  0xae
. byte   0x64
.uleb128 0xc
.4 byte  $LASF42
. byte   0x5
.2 byte  0x12e
.4 byte  0x25
. byte   0x68
.uleb128 0xc
.4 byte  $LASF43
. byte   0x5
.2 byte  0x12f
.4 byte  0x5a
. byte   0x6c
.uleb128 0xc
.4 byte  $LASF44
. byte   0x5
.2 byte  0x131
.4 byte  0x2a4
. byte   0x70
. byte   0
.uleb128 0xd
.4 byte  $LASF91
. byte   0x5
. byte   0x96
.uleb128 0xa
.4 byte  $LASF46
. byte   0xc
. byte   0x5
. byte   0x9c
.4 byte  0x282
.uleb128 0xb
.4 byte  $LASF47
. byte   0x5
. byte   0x9d
.4 byte  0x282
. byte   0
.uleb128 0xb
.4 byte  $LASF48
. byte   0x5
. byte   0x9e

```

```
.4 byte 0x288
. byte 0x4
. uleb128 0xb
.4 byte $LASF49
. byte 0x5
. byte 0xa2
.4 byte 0x5a
. byte 0x8
. byte 0
. uleb128 0x8
. byte 0x4
.4 byte 0x251
. uleb128 0x8
. byte 0x4
.4 byte 0xcd
. uleb128 0x5
.4 byte 0xb6
.4 byte 0x29e
. uleb128 0x6
.4 byte 0xa7
. byte 0
. byte 0
. uleb128 0x8
. byte 0x4
.4 byte 0x24a
. uleb128 0x5
.4 byte 0xb6
.4 byte 0x2b4
. uleb128 0x6
.4 byte 0xa7
. byte 0x27
. byte 0
. uleb128 0xe
.4 byte $LASF92
. uleb128 0xf
.4 byte $LASF50
. byte 0x5
.2 byte 0x13b
.4 byte 0x2b4
. uleb128 0xf
.4 byte $LASF51
. byte 0x5
.2 byte 0x13c
.4 byte 0x2b4
. uleb128 0xf
.4 byte $LASF52
. byte 0x5
.2 byte 0x13d
.4 byte 0x2b4
. uleb128 0x8
. byte 0x4
.4 byte 0xbd
. uleb128 0x9
.4 byte 0x2dd
. uleb128 0x10
.4 byte $LASF53
```

```
.byte    0x4
.byte    0xaa
.4byte   0x288
.uleb128 0x10
.4byte   $LASF54
.byte    0x4
.byte    0xab
.4byte   0x288
.uleb128 0x10
.4byte   $LASF55
.byte    0x4
.byte    0xac
.4byte   0x288
.uleb128 0x10
.4byte   $LASF56
.byte    0x6
.byte    0x1a
.4byte   0x5a
.uleb128 0x5
.4byte   0x2e3
.4byte   0x31f
.uleb128 0x11
.byte    0
.uleb128 0x9
.4byte   0x314
.uleb128 0x10
.4byte   $LASF57
.byte    0x6
.byte    0x1b
.4byte   0x31f
.uleb128 0x10
.4byte   $LASF58
.byte    0x7
.byte    0x39
.4byte   0xb0
.uleb128 0x10
.4byte   $LASF59
.byte    0x7
.byte    0x47
.4byte   0x5a
.uleb128 0x10
.4byte   $LASF60
.byte    0x7
.byte    0x4c
.4byte   0x5a
.uleb128 0x10
.4byte   $LASF61
.byte    0x7
.byte    0x50
.4byte   0x5a
.uleb128 0x8
.byte    0x4
.4byte   0x5a
.uleb128 0x3
.byte    0x8
.byte    0x4
```

```

.4byte $LASF62
.uleb128 0x12
.ascii "gcd\000"
.byte 0xc
.byte 0x8
.byte 0x6
.4byte 0x399
.uleb128 0xb
.4byte $LASF63
.byte 0x8
.byte 0x7
.4byte 0x5a
.byte 0
.uleb128 0xb
.4byte $LASF64
.byte 0x8
.byte 0x8
.4byte 0x5a
.byte 0x4
.uleb128 0xb
.4byte $LASF65
.byte 0x8
.byte 0x9
.4byte 0x5a
.byte 0x8
.byte 0
.uleb128 0x13
.4byte $LASF67
.byte 0x1
.byte 0x97
.4byte 0x3c1
.4byte $LFB8
.4byte $LFE8-$LFB8
.uleb128 0x1
.byte 0x9c
.4byte 0x3c1
.uleb128 0x14
.4byte $LASF69
.byte 0x1
.byte 0x97
.4byte 0xb6
.uleb128 0x2
.byte 0x91
.sleb128 0
.byte 0
.uleb128 0x3
.byte 0x1
.byte 0x2
.4byte $LASF66
.uleb128 0x15
.4byte $LASF68
.byte 0x1
.byte 0x93
.4byte 0x3c1
.4byte $LFB7
.4byte $LFE7-$LFB7

```

```

.uleb128 0x1
.byte 0x9c
.4byte 0x3f0
.uleb128 0x14
.4byte $LASF69
.byte 0x1
.byte 0x93
.4byte 0xb6
.uleb128 0x2
.byte 0x91
.sleb128 0
.byte 0
.uleb128 0x15
.4byte $LASF70
.byte 0x1
.byte 0x8f
.4byte 0x3c1
.4byte $LFB6
.4byte $LFE6-$LFB6
.uleb128 0x1
.byte 0x9c
.4byte 0x418
.uleb128 0x14
.4byte $LASF69
.byte 0x1
.byte 0x8f
.4byte 0xb6
.uleb128 0x2
.byte 0x91
.sleb128 0
.byte 0
.uleb128 0x16
.4byte $LASF82
.byte 0x1
.byte 0x87
.4byte $LFB5
.4byte $LFE5-$LFB5
.uleb128 0x1
.byte 0x9c
.4byte 0x46e
.uleb128 0x17
.ascii "gcd\000"
.byte 0x1
.byte 0x87
.4byte 0x46e
.uleb128 0x2
.byte 0x91
.sleb128 0
.uleb128 0x14
.4byte $LASF71
.byte 0x1
.byte 0x87
.4byte 0x25
.uleb128 0x2
.byte 0x91
.sleb128 4

```

```

.uleb128 0x14
.4byte $LASF72
.byte 0x1
.byte 0x87
.4byte 0x474
.uleb128 0x2
.byte 0x91
.sleb128 8
.uleb128 0x18
.4byte $LBB2
.4byte $LBE2-$LBB2
.uleb128 0x19
.ascii "i\000"
.byte 0x1
.byte 0x8a
.4byte 0x5a
.uleb128 0x2
.byte 0x91
.sleb128 -16
.byte 0
.byte 0
.uleb128 0x8
.byte 0x4
.4byte 0x368
.uleb128 0x8
.byte 0x4
.4byte 0xc2
.uleb128 0x13
.4byte $LASF73
.byte 0x1
.byte 0x6e
.4byte 0x5a
.4byte $LFB4
.4byte $LFE4-$LFB4
.uleb128 0x1
.byte 0x9c
.4byte 0x501
.uleb128 0x14
.4byte $LASF74
.byte 0x1
.byte 0x6e
.4byte 0xb0
.uleb128 0x2
.byte 0x91
.sleb128 0
.uleb128 0x14
.4byte $LASF75
.byte 0x1
.byte 0x6e
.4byte 0x5a
.uleb128 0x2
.byte 0x91
.sleb128 4
.uleb128 0x14
.4byte $LASF76
.byte 0x1

```

```

.byte    0x6e
.4byte   0x35b
.uleb128 0x2
.byte    0x91
.sleb128 8
.uleb128 0x1a
.4byte   $LASF77
.byte    0x1
.byte    0x6f
.4byte   0x501
.uleb128 0x3
.byte    0x91
.sleb128 -24
.byte    0x6
.uleb128 0x1a
.4byte   $LASF69
.byte    0x1
.byte    0x70
.4byte   0xb6
.uleb128 0x2
.byte    0x91
.sleb128 -20
.uleb128 0x1a
.4byte   $LASF78
.byte    0x1
.byte    0x71
.4byte   0x5a
.uleb128 0x2
.byte    0x91
.sleb128 -40
.uleb128 0x19
.ascii   "i\000"
.byte    0x1
.byte    0x73
.4byte   0x5a
.uleb128 0x2
.byte    0x91
.sleb128 -36
.uleb128 0x19
.ascii   "j\000"
.byte    0x1
.byte    0x74
.4byte   0x5a
.uleb128 0x2
.byte    0x91
.sleb128 -32
.byte    0
.uleb128 0x5
.4byte   0xb6
.4byte   0x514
.uleb128 0x1b
.4byte   0xa7
.uleb128 0x3
.byte    0x91
.sleb128 -28
.byte    0x6

```



```

.byte    0
.uleb128 0x13
.4byte   $LASF79
.byte    0x1
.byte    0x44
.4byte   0x5a
.4byte   $LFB3
.4byte   $LFE3-$LFB3
.uleb128 0x1
.byte    0x9c
.4byte   0x574
.uleb128 0x14
.4byte   $LASF80
.byte    0x1
.byte    0x44
.4byte   0x474
.uleb128 0x2
.byte    0x91
.sleb128 0
.uleb128 0x14
.4byte   $LASF75
.byte    0x1
.byte    0x44
.4byte   0x35b
.uleb128 0x2
.byte    0x91
.sleb128 4
.uleb128 0x14
.4byte   $LASF74
.byte    0x1
.byte    0x44
.4byte   0x574
.uleb128 0x2
.byte    0x91
.sleb128 8
.uleb128 0x1a
.4byte   $LASF81
.byte    0x1
.byte    0x45
.4byte   0x5a
.uleb128 0x2
.byte    0x91
.sleb128 -16
.uleb128 0x1a
.4byte   $LASF69
.byte    0x1
.byte    0x4d
.4byte   0x5a
.uleb128 0x2
.byte    0x91
.sleb128 -12
.byte    0
.uleb128 0x8
.byte    0x4
.4byte   0xb0
.uleb128 0x1c

```

```

.4byte $LASF83
.byte 0x1
.byte 0x8
.4byte 0x5a
.4byte $LFB2
.4byte $LFE2-$LFB2
.uleb128 0x1
.byte 0x9c
.uleb128 0x14
.4byte $LASF84
.byte 0x1
.byte 0x8
.4byte 0x474
.uleb128 0x2
.byte 0x91
.sleb128 0
.uleb128 0x14
.4byte $LASF72
.byte 0x1
.byte 0x8
.4byte 0x474
.uleb128 0x2
.byte 0x91
.sleb128 4
.uleb128 0x1a
.4byte $LASF85
.byte 0x1
.byte 0xa
.4byte 0x5a
.uleb128 0x2
.byte 0x91
.sleb128 -40
.uleb128 0x1a
.4byte $LASF76
.byte 0x1
.byte 0xb
.4byte 0x97
.uleb128 0x2
.byte 0x91
.sleb128 -24
.uleb128 0x1a
.4byte $LASF75
.byte 0x1
.byte 0xc
.4byte 0x5a
.uleb128 0x2
.byte 0x91
.sleb128 -16
.uleb128 0x1a
.4byte $LASF74
.byte 0x1
.byte 0xd
.4byte 0xb0
.uleb128 0x2
.byte 0x91
.sleb128 -12

```

```

.uleb128 0x1a
.4byte $LASF81
.byte 0x1
.byte 0xf
.4byte 0x5a
.uleb128 0x2
.byte 0x91
.sleb128 -36
.uleb128 0x1a
.4byte $LASF86
.byte 0x1
.byte 0x10
.4byte 0x25
.uleb128 0x2
.byte 0x91
.sleb128 -32
.uleb128 0x1a
.4byte $LASF87
.byte 0x1
.byte 0x11
.4byte 0x46e
.uleb128 0x2
.byte 0x91
.sleb128 -28
.byte 0
.byte 0
.section .debug_abbrev,"",@progbits
$Ldebug_abbrev0:
.uleb128 0x1
.uleb128 0x11
.byte 0x1
.uleb128 0x25
.uleb128 0xe
.uleb128 0x13
.uleb128 0xb
.uleb128 0x3
.uleb128 0xe
.uleb128 0x1b
.uleb128 0xe
.uleb128 0x11
.uleb128 0x1
.uleb128 0x12
.uleb128 0x6
.uleb128 0x10
.uleb128 0x17
.byte 0
.byte 0
.uleb128 0x2
.uleb128 0x16
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb

```

```
.uleb128 0x49
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x3
.uleb128 0x24
.byte 0
.uleb128 0xb
.uleb128 0xb
.uleb128 0x3e
.uleb128 0xb
.uleb128 0x3
.uleb128 0xe
.byte 0
.byte 0
.uleb128 0x4
.uleb128 0x24
.byte 0
.uleb128 0xb
.uleb128 0xb
.uleb128 0x3e
.uleb128 0xb
.uleb128 0x3
.uleb128 0x8
.byte 0
.byte 0
.uleb128 0x5
.uleb128 0x1
.byte 0x1
.uleb128 0x49
.uleb128 0x13
.uleb128 0x1
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x6
.uleb128 0x21
.byte 0
.uleb128 0x49
.uleb128 0x13
.uleb128 0x2f
.uleb128 0xb
.byte 0
.byte 0
.uleb128 0x7
.uleb128 0xf
.byte 0
.uleb128 0xb
.uleb128 0xb
.byte 0
.byte 0
.uleb128 0x8
.uleb128 0xf
.byte 0
.uleb128 0xb
.uleb128 0xb
```

```
.uleb128 0x49
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x9
.uleb128 0x26
.byte 0
.uleb128 0x49
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0xa
.uleb128 0x13
.byte 0x1
.uleb128 0x3
.uleb128 0xe
.uleb128 0xb
.uleb128 0xb
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x1
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0xb
.uleb128 0xd
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.uleb128 0x38
.uleb128 0xb
.byte 0
.byte 0
.uleb128 0xc
.uleb128 0xd
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0x5
.uleb128 0x49
.uleb128 0x13
.uleb128 0x38
.uleb128 0xb
.byte 0
.byte 0
```

```
.uleb128 0xd
.uleb128 0x16
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.byte 0
.byte 0
.uleb128 0xe
.uleb128 0x13
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3c
.uleb128 0x19
.byte 0
.byte 0
.uleb128 0xf
.uleb128 0x34
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0x5
.uleb128 0x49
.uleb128 0x13
.uleb128 0x3f
.uleb128 0x19
.uleb128 0x3c
.uleb128 0x19
.byte 0
.byte 0
.uleb128 0x10
.uleb128 0x34
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.uleb128 0x3f
.uleb128 0x19
.uleb128 0x3c
.uleb128 0x19
.byte 0
.byte 0
.uleb128 0x11
.uleb128 0x21
```

```
.byte 0
.byte 0
.byte 0
.uleb128 0x12
.uleb128 0x13
.byte 0x1
.uleb128 0x3
.uleb128 0x8
.uleb128 0xb
.uleb128 0xb
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x1
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x13
.uleb128 0x2e
.byte 0x1
.uleb128 0x3f
.uleb128 0x19
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x27
.uleb128 0x19
.uleb128 0x49
.uleb128 0x13
.uleb128 0x11
.uleb128 0x1
.uleb128 0x12
.uleb128 0x6
.uleb128 0x40
.uleb128 0x18
.uleb128 0x2116
.uleb128 0x19
.uleb128 0x1
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x14
.uleb128 0x5
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
```

```
.uleb128 0x2
.uleb128 0x18
.byte 0
.byte 0
.uleb128 0x15
.uleb128 0x2e
.byte 0x1
.uleb128 0x3f
.uleb128 0x19
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x27
.uleb128 0x19
.uleb128 0x49
.uleb128 0x13
.uleb128 0x11
.uleb128 0x1
.uleb128 0x12
.uleb128 0x6
.uleb128 0x40
.uleb128 0x18
.uleb128 0x2117
.uleb128 0x19
.uleb128 0x1
.uleb128 0x13
.byte 0
.byte 0
.uleb128 0x16
.uleb128 0x2e
.byte 0x1
.uleb128 0x3f
.uleb128 0x19
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x27
.uleb128 0x19
.uleb128 0x11
.uleb128 0x1
.uleb128 0x12
.uleb128 0x6
.uleb128 0x40
.uleb128 0x18
.uleb128 0x2116
.uleb128 0x19
.uleb128 0x1
.uleb128 0x13
.byte 0
.byte 0
```



```
.uleb128 0x17
.uleb128 0x5
.byte 0
.uleb128 0x3
.uleb128 0x8
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.uleb128 0x2
.uleb128 0x18
.byte 0
.byte 0
.uleb128 0x18
.uleb128 0xb
.byte 0x1
.uleb128 0x11
.uleb128 0x1
.uleb128 0x12
.uleb128 0x6
.byte 0
.byte 0
.uleb128 0x19
.uleb128 0x34
.byte 0
.uleb128 0x3
.uleb128 0x8
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.uleb128 0x2
.uleb128 0x18
.byte 0
.byte 0
.uleb128 0x1a
.uleb128 0x34
.byte 0
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.uleb128 0x49
.uleb128 0x13
.uleb128 0x2
.uleb128 0x18
.byte 0
.byte 0
.uleb128 0x1b
.uleb128 0x21
```

```

        .byte    0
        .uleb128 0x49
        .uleb128 0x13
        .uleb128 0x2f
        .uleb128 0x18
        .byte    0
        .byte    0
        .uleb128 0x1c
        .uleb128 0x2e
        .byte    0x1
        .uleb128 0x3f
        .uleb128 0x19
        .uleb128 0x3
        .uleb128 0xe
        .uleb128 0x3a
        .uleb128 0xb
        .uleb128 0x3b
        .uleb128 0xb
        .uleb128 0x27
        .uleb128 0x19
        .uleb128 0x49
        .uleb128 0x13
        .uleb128 0x11
        .uleb128 0x1
        .uleb128 0x12
        .uleb128 0x6
        .uleb128 0x40
        .uleb128 0x18
        .uleb128 0x2116
        .uleb128 0x19
        .byte    0
        .byte    0
        .byte    0
        .section          .debug_aranges,"",@progbits
        .4byte    0x1c
        .2byte    0x2
        .4byte    $Ldebug_info0
        .byte     0x4
        .byte     0
        .2byte    0
        .2byte    0
        .4byte    $Ltext0
        .4byte    $Ltext0-$Ltext0
        .4byte    0
        .4byte    0
        .section          .debug_line,"",@progbits
$Ldebug_line0:
        .section          .debug_str,"MS",@progbits,1
$LASF10:
        .ascii    "__off_t\000"
$LASF17:
        .ascii    "_IO_read_ptr\000"
$LASF29:
        .ascii    "_chain\000"
$LASF82:
        .ascii    "imprimir_salida\000"

```

```

$LASF8:
. ascii  "size_t\000"
$LASF83:
. ascii  "procesar_archivos\000"
$LASF35:
. ascii  "_shortbuf\000"
$LASF72:
. ascii  "salida\000"
$LASF80:
. ascii  "stream\000"
$LASF52:
. ascii  "_IO_2_1_stderr_\000"
$LASF23:
. ascii  "_IO_buf_base\000"
$LASF7:
. ascii  "long_long_unsigned_int\000"
$LASF77:
. ascii  "temporal\000"
$LASF6:
. ascii  "long_long_int\000"
$LASF4:
. ascii  "signed_char\000"
$LASF86:
. ascii  "largo_arreglo\000"
$LASF70:
. ascii  "es_fin_de_linea\000"
$LASF30:
. ascii  "_fileno\000"
$LASF78:
. ascii  "largo_enteros\000"
$LASF18:
. ascii  "_IO_read_end\000"
$LASF11:
. ascii  "long_int\000"
$LASF16:
. ascii  "_flags\000"
$LASF24:
. ascii  "_IO_buf_end\000"
$LASF33:
. ascii  "_cur_column\000"
$LASF9:
. ascii  "__quad_t\000"
$LASF63:
. ascii  "num_a\000"
$LASF32:
. ascii  "_old_offset\000"
$LASF37:
. ascii  "_offset\000"
$LASF67:
. ascii  "es_caracter_invalido\000"
$LASF62:
. ascii  "long_double\000"
$LASF46:
. ascii  "_IO_marker\000"
$LASF53:
. ascii  "stdin\000"

```

```

$LASF0:      .ascii  "unsigned_int\000"
$LASF3:      .ascii  "long_unsigned_int\000"
$LASF92:     .ascii  "_IO_FILE_plus\000"
$LASF89:     .ascii  "max_divisor.c\000"
$LASF21:     .ascii  "_IO_write_ptr\000"
$LASF74:     .ascii  "linea\000"
$LASF56:     .ascii  "sys_nerr\000"
$LASF48:     .ascii  "_sbuf\000"
$LASF2:      .ascii  "short_unsigned_int\000"
$LASF25:     .ascii  "_IO_save_base\000"
$LASF36:     .ascii  "_lock\000"
$LASF31:     .ascii  "_flags2\000"
$LASF43:     .ascii  "_mode\000"
$LASF54:     .ascii  "stdout\000"
$LASF73:     .ascii  "pasar_a_enteros\000"
$LASF50:     .ascii  "_IO_2_1_stdin_\000"
$LASF87:     .ascii  "arreglo_structs\000"
$LASF58:     .ascii  "optarg\000"
$LASF88:     .ascii  "GNU_C11_6.3.0_20170516_-meb_-march=mips32r2_-mfpxx_-mlls"
             .ascii  "c_-mno-lxc1-sxc1_-mips32r2_-mabi=32_-g\000"
$LASF13:     .ascii  "sizetype\000"
$LASF90:     .ascii  "/home/manu/TP1/OrgaDeCompus-TP1/Resolucion\000"
$LASF59:     .ascii  "optind\000"
$LASF22:     .ascii  "_IO_write_end\000"
$LASF68:     .ascii  "es_numerico\000"
$LASF91:     .ascii  "_IO_lock_t\000"
$LASF45:     .ascii  "_IO_FILE\000"
$LASF69:     .ascii  "caracter\000"
$LASF49:

```

```

        .ascii  "_pos\000"
$LASF57:
        .ascii  "sys_errlist\000"
$LASF28:
        .ascii  "_markers\000"
$LASF76:
        .ascii  "enteros\000"
$LASF66:
        .ascii  "_Bool\000"
$LASF1:
        .ascii  "unsigned_char\000"
$LASF5:
        .ascii  "short_int\000"
$LASF79:
        .ascii  "leer_linea\000"
$LASF34:
        .ascii  "_vtable_offset\000"
$LASF64:
        .ascii  "num_b\000"
$LASF51:
        .ascii  "_IO_2_1_stdout_\000"
$LASF15:
        .ascii  "FILE\000"
$LASF61:
        .ascii  "optopt\000"
$LASF65:
        .ascii  "gcd_ab\000"
$LASF14:
        .ascii  "char\000"
$LASF84:
        .ascii  "entrada\000"
$LASF60:
        .ascii  "opterr\000"
$LASF47:
        .ascii  "_next\000"
$LASF12:
        .ascii  "__off64_t\000"
$LASF71:
        .ascii  "largo\000"
$LASF19:
        .ascii  "_IO_read_base\000"
$LASF27:
        .ascii  "_IO_save_end\000"
$LASF81:
        .ascii  "largo_buffer\000"
$LASF38:
        .ascii  "--pad1\000"
$LASF39:
        .ascii  "--pad2\000"
$LASF40:
        .ascii  "--pad3\000"
$LASF41:
        .ascii  "--pad4\000"
$LASF42:
        .ascii  "--pad5\000"
$LASF44:

```

```

        . a s c i i   " _ u n u s e d 2 \ 0 0 0 "
$LASF55:
        . a s c i i   " s t d e r r \ 0 0 0 "
$LASF75:
        . a s c i i   " l a r g o _ l i n e a \ 0 0 0 "
$LASF26:
        . a s c i i   " _ I O _ b a c k u p _ b a s e \ 0 0 0 "
$LASF20:
        . a s c i i   " _ I O _ w r i t e _ b a s e \ 0 0 0 "
$LASF85:
        . a s c i i   " l e c t u r a \ 0 0 0 "
        . i d e n t   " G C C : _ ( D e b i a n _ 6 . 3 . 0 - 1 8 + d e b 9 u 1 ) _ 6 . 3 . 0 _ 2 0 1 7 0 5 1 6 "

```