Trabajo Práctico 1 — MIPS32
GCD : Algoritmo de Euclides

[66.20] Organización del Computador
Curso Martes
Primer cuatrimestre de 2021

Vázquez Lareu, Román	100815
Movia, Guido	102896
Meza Boeykens, Damián	102696

Índice

1. Introducción

En este informe tenemos como objetivo desarrollar un programa el cual implementará el algoritmo de Euclides sobre una cantidad arbitraria de pares de números, con el fin de familiarizarnos con el conjunto de instrucciones MIPS y el concepto de ABI.

El algoritmo de euclides es un método que se utiliza para calcular el máximo común divisor (GCD) entre dos o más números enteros, el GCD se define como el mayor número entero que divide a los argumentos sin dejar un resto. Al dividir un número entero a entre un número entero b se obtiene un cociente q y un resto r. A partir de ello se puede demostrar que el máximo común divisor entre a y b es equivalente al máximo común divisor entre b y r, y éste es el principio fundamental del algoritmo, el cual se encarga de realizar iteraciones hasta que el segundo argumento sea nulo , de forma tal que el GCD estara compuesto por el primer argumento. De esta manera se prosigue hasta lograr satisfacer la siguiente ecuación:

$$r_{k-2} = q_k \cdot r_{k-1} + r_k \tag{1}$$

donde k representa el paso, r el resto y q el cociente.

2. Diseño e implementación del programa

2.1. Programa principal

El programa principal, en lenguaje C, se encarga de recibir los pares de números a calcular ya sea por archivo txt o por *stdin*, alocar en memoria cada uno de estos en un bloque contiguo y llamar a la función externa *euclides*. Luego imprime por la salida que corresponda cada uno de los resultados.

2.2. Función Euclides

La función Euclides, en lenguaje Assembly, es la función externa y es la que puede ser llamada por otras, tanto las regresiones como el programa principal. Recibe por parámetros el puntero al inicio del arreglo y el tamaño del arreglo, luego se encarga de llamar a la función GCD la cual recibe como argumento dos números enteros los cuales provienen de las primeras dos posiciones del struct. La función GCD se encarga de calcular el máximo común divisor entre ambos números de forma recursiva y utilizando el algoritmo de euclides. Por último,la función euclides retoma este valor y lo escribe en la última posición del struct. Este procedimiento se repite consecutivamente a traves de un ciclo while, el cual finaliza cuando hayamos recorrido cada uno de los struct.

2.2.1. Explicación del código Assembly

Función Euclides

Las primeras líneas se corresponden con la iniciación y construcción del stack frame. Luego se guardan los párametros pasados por la funcion caller. Se salva la dirección recibida por párametro para poder manipularla, se guarda el primer valor del struct y luego el segundo (se accede a a0 con un offset) Luego de un chequeo por cero, se trasladan los enteros a su valor absoluto para llamar a la función GCD, si alguno es cero la función saltea el cálculo del GCD y escribe un cero como resultado. A continuación, de acuerdo a si el primero es mayor o menor al segundo se hace el intercambio y se llama a la función GCD ya que ésta necesita si o si que el primero sea mayor o igual al segundo. Podemos observar que el stack frame de la funcion Euclides se encuentra compuesto por el SRA (General Register Save Area) en la cual se encarga por notacion de salvar a los registros fp, gp y ra (este ultimo se salva ya que la funcion es non-leaf), la LTA (Local and Temporary Variables Area) la cual se encarga de almacenar las variables locales, que en este caso estan dadas por la direccion del struct y su tamaño, mientras que la ABA (Argument Building Area) almacena los argumentos de la funcion que sera llamada (GCD), siendo estos los numeros enteros a y b.

Función GCD

La funcion GCD inicia y construye su stack frame de forma analoga al caso anterior. A partir de ello se cargan los argumentos en los registros temporales t0 y t1, y se realiza una division entre ambos resultados, de forma tal que el resto quedara almacenado en el registro hi. El algoritmo de euclides nos afirma que el GCD entre a y b es equivalente al GCD entre b y el resto de la division, por lo tanto lo que haremos sera almacenar en el registro t0 que contenia anteriormente a la variable a el contenido del registro t1, y esto lo hacemos con una instruccion move, mientras que para almacenar el resto en el registro t1 lo que haremos sera utilizar la operacion mfhi. Luego se verifica que el resto sea nulo, y en ese caso el GCD estara dado por el valor que se encuentra en el registro t0, en caso contrario se llama nuevamente a la funcion con sus nuevos parametros. Finalmente se elimina el stack frame y el salto al registro ra nos lleva a la direccion siguiente al llamado recursivo anterior, por lo que de esta manera se va destruyendo el stack frame hasta que finalmente se vuelve a la funcion que hizo la llamada original. El stack frame de la funcion GCD posee analogamente las secciones SRA, LTA y ABA. En la primera se salvan los registros fp, gp y ra (igual al caso anterior ya que la funcion por ser recursiva es non-leaf), en la segunda se almacenan las variables locales que estan dadas por los argumentos que recibe la funcion (los cuales son modificados a lo largo de la funcion) y en la tercera se almacenaran los argumentos de la funcion que sera llamada, la cual sera nuevamente la funcion GCD pero a diferencia de la anterior, los argumentos son la variable b y el resto de la division generada anteriormente.

2.2.2. Diagrama de los StackFrames

Stack Frame de Euclides		Stack Frame de GCD	
44 n	ABA CALLER	44 b	ABA CALLER
40 dir	ABA CALLER	40 a	ABA CALLER
36	SRA	36	SRA
32 ra	SRA	32 ra	SRA
28 fp	SRA	28 fp	SRA
24 gp	SRA	24 gp	SRA
20 n	LTA	20 a	LTA
16 dir	LTA	16 b	LTA
12	ABA	12	ABA
8	ABA	8	ABA
4 b	ABA	4 r	ABA
0 a	ABA	0 b	ABA

Figura 1: StackFrame de ambas funciones

2.3. Comandos para compilar el programa

Ejecutando el comando *make* dentro de la carpeta de este TP, se compilarán tanto las regresiones como el programa tp1.

```
root@debian-stretch-mips:~/tmp/tp# make
cc -Wall -g -o regressions regressions.c gcd.S
cc -Wall -g -o tp1 tp1.c gcd.S
:
root@debian-stretch-mips:~/tmp/tp#
```

Figura 2: Compilación mediante el comando make

Para compilar sólamente el programa se debe tener el archivo gcd.S, gcd.h y tp1.c, luego ejecutar en la terminal:

```
gcc tp1.c gcd.S -o tp1
```

2.4. Corridas de prueba

Mostramos como ejemplo en las figuras 3, 4, 5 y 6, teniendo ya compilado el programa tp1, la ejecución de las siguientes pruebas:

```
root@debian-stretch-mips:~/tmp/tp# cat vacio.txt
root@debian-stretch-mips:~/tmp/tp# ./tp1 -i vacio.txt
root@debian-stretch-mips:~/tmp/tp# ./tp1 -i vacio.txt -o salida.txt
root@debian-stretch-mips:~/tmp/tp# cat salida.txt
root@debian-stretch-mips:~/tmp/tp#
```

Figura 3: Corridas de prueba con entrada vacía

```
GCD(987,444) = 3
GCD(988,444) = 4
GCD(989,444) = 1
GCD(990,444) = 6
GCD(991,444) = 1
GCD(992,444) = 4
GCD(993,444) = 3
GCD(994,444) = 2
GCD(995,444) = 1
GCD(996,444) = 12
GCD(997,444) = 1
GCD(998,444) = 2
GCD(999,444) = 111
GCD(1000,444) = 4
root@debian-stretch-mips:~/tmp/tp#
```

Figura 4: Corridas de prueba con múltiples líneas y salida por consola

Figura 5: Corridas de prueba con un cero en la entrada

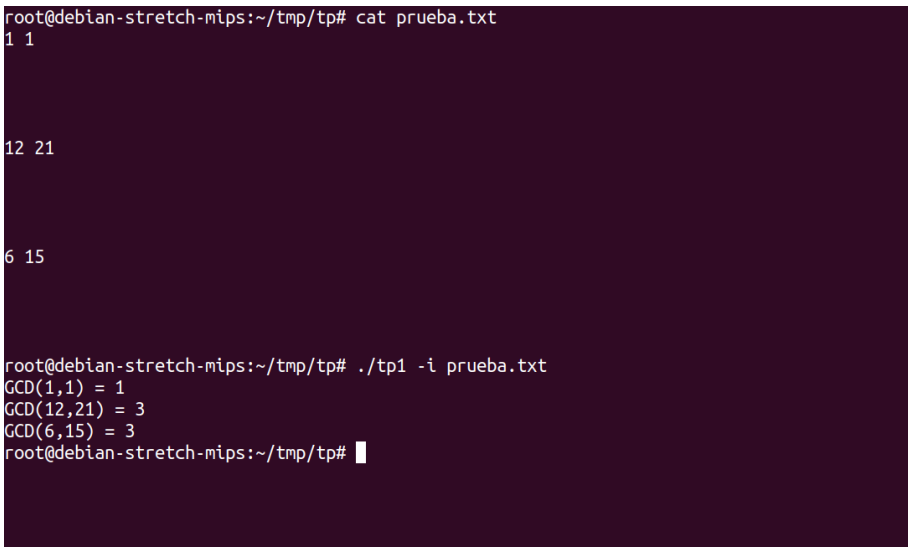

Figura 6: Corridas de prueba con saltos de línea en la entrada

3. Códigos fuentes

3.1. Código fuente Assembly

```
        .abicalls
        .align 2
        .text
        .globl euclides
        .ent euclides

euclides:
        .frame        $fp,40,$ra
        subu        $sp,$sp,40
        .cprestore 24
        sw        $ra,32($sp)
        sw        $fp,28($sp)
        move        $fp,$sp

        sw        $a0,40($fp)                    # ABA
        sw        $a1,44($fp)

        sw        $a0,20($fp)                 # LTA
        sw        $a1,16($fp)


        lw        $t4,20($fp)              #Guardo dir del arreglo
        lw        $t5,16($fp)             # Guardo tamaño de arreglo

while: lw        $t2,($t4)               #Guardo A

        lw        $t3,4($t4)              #Guardo B

        li        $v0,0
        beq        $t2,$zero,seguir        # Si alguno es 0 -> MCD = 0
        beq        $t3,$zero,seguir

        abs        $t2,$t2
        abs        $t3,$t3

        blt        $t2,$t3,swap
noswap: move        $a0,$t2
        move        $a1,$t3
        b llamada

swap: move        $a0,$t3
        move        $a1,$t2

llamada: jal gcd

seguir: sw        $v0,8($t4)

        addu        $t4,$t4,12        #dir offset
        addu        $t5,$t5,-1        #size-1
        beq        $t5,$zero,fin
```

```
        b while

gcd:
        .frame        $fp,40,$ra
        subu        $sp,$sp,40
        .cprestore 24
        sw        $ra,32($sp)
        sw        $fp,28($sp)
        move        $fp,$sp

        sw        $a0,40($fp)                          # ABA
        sw        $a1,44($fp)

        sw        $a0,20($fp)                          # LTA
        sw        $a1,16($fp)

        lw        $t0,20($fp)                           # Almaceno la variable r1 en el registro t
        lw        $t1,16($fp)                           # Almaceno la variable r2 en el registro t
        div        $t0,$t1                     # Divido el contenido del registro t0 (r1) con el c
        move        $t0,$t1                        # Copio el contenido del registro t1 (r2) en el
        mfhi        $t1                             # Almaceno el resto de la division realizad
        beq        $t1,$zero,igualA0              # if(r2 == 0) entonces el MCD es r1
        addu        $a0,$t0,$zero                  # Almaceno el contenido de t0 (r1) en el regis
        addu        $a1,$t1,$zero                  # Almaceno el contenido de t1 (r2) en el regis
        jal gcd
        b fin

igualA0: lw        $v0,16($fp)                      # Almaceno en el registro de retorno v0 el co

fin:        lw        $fp,28($sp)
        lw        $ra,32($sp)
        addiu        $sp,$sp,40
        jr $ra

        .end euclides
```

3.2. Código fuente C del programa principal

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gcd.h"

#define VERSION 1.2

void mensaje_ayuda() {
          printf("Usage: \n");
        printf("        tp1 -h \n");
        printf("        tp1 -V \n");
        printf("        tp1 -i in_file -o out_file \n");
        printf("Options: \n ");
        printf("        -V, --version     Print version and quit.\n" );
```

```
        printf("        -h, --help        Print this information and quit. \n");
        printf("        -i, --input       Specify input stream/file, '-' for stdin. \n");
        printf("        -o, --output      Specify output stream/file, '-' for stdout \n");
        printf("Examples: \n");
        printf("        tp1 < in.txt > out.txt \n");
        printf("        cat in.txt | tp1 -i - > out.txt \n");
}

int main(int argc,char* argv[]){
        char* input;
        char* output;
        int modo_out=0;        // 0: stdout | 1: file
        int modo_in=0;                // 0: stdin  | 1: file

        int i;

        FILE * file_in=stdin;
        FILE * file_out=stdout;

        for (i=1;i<argc;i++) {

                //HELP
                if (strcmp(argv[i], "-h") == 0) {
                        mensaje_ayuda();
                        return 0;
                }
                //VERSION
                if (strcmp(argv[i], "-V") == 0) {
                        printf("Versión del TP: %.1f \n", VERSION);
                        return 0;
                }
                //IN
                if (strcmp(argv[i], "-i")==0) {

                        if (argc>=(i+1)) {

                                if (strcmp(argv[i+1], "-")==0) {
                                        modo_in = 0;
                                }
                                else {
                                        modo_in = 1;
                                        input = argv[i+1];
                                }
                        }
                }
                //OUT
                if (strcmp(argv[i], "-o")==0) {

                        if (argc>=(i+1)) {

                                if (strcmp(argv[i+1], "-")==0) {
                                        modo_out = 0;
                                }
                                else {
```

```
                                modo_out = 1;
                                output = argv[i+1];
                        }
                }
        }
}


if (modo_in==1) {
        file_in = fopen(input,"r");
        //verifico que abri bien el archivo
        if (file_in==NULL){
                return -1;
        }
}

if(modo_out==1) {
        file_out = fopen(output,"w");
        //verifico que abri bien el archivo
        if (file_out==NULL){
                return -1;
        }
}

struct gcd* puntero=NULL;
size_t largo_array=0;
//creo buffer para almacenar linea (max 100 caracteres)
char buffer[100];

if (feof(file_in)){
        return 0;
}
//leo la primer linea
fgets(buffer, 100, file_in);
while(!feof(file_in)){
        struct gcd p;
        //ignoro saltos de línea
        if (buffer[0]!='\n') {
                //leo linea y asigno lo que lea a posiciones del struct
                sscanf(buffer,"%d %d", &p.num_a, &p.num_b);

                struct gcd* try = realloc(puntero, (largo_array+1) * sizeof(struct gcd));
                if(try==NULL) { // SIN MEMORIA
                        printf("Memory Alloc Error\n");
                        return -1;
                }
                puntero = try;
                puntero[largo_array] = p;
                largo_array++;
        }
        //leo siguiente linea
        fgets(buffer,100, file_in);
}
```

```
        if (largo_array == 0) {
                if(modo_in==1) {
                        fclose(file_in);
                }
                if(modo_out==1) {
                        fclose(file_out);
                }
                return -1;
        }

        //LLAMO A EUCLIDES y le paso puntero al array y largo_array
        euclides(puntero,largo_array);

        //recorro el struct para printear la salida
        for (i=0;i<largo_array;i++){
                fprintf(file_out, "GCD(%d,%d) = %d \n", puntero[i].num_a, puntero[i].num_b, punte
        }
        //Cierro archivos
        if(modo_in==1) {
                fclose(file_in);
        }
        if(modo_out==1) {
                fclose(file_out);
        }
        if(puntero!=NULL) {
                free(puntero);
        }
        return 0;
}
```

3.3. Código MIPS32 generado por el compilador

```
        .section .mdebug.abi32
        .previous
        .nan        legacy
        .module        fp=xx
        .module        nooddspreg
        .abicalls
        .text
$Ltext0:
        .cfi_sections        .debug_frame
        .rdata
        .align        2
$LC0:
        .ascii        "Usage: \000"
        .align        2
$LC1:
        .ascii        "\011tp1 -h \000"
        .align        2
$LC2:
        .ascii        "\011tp1 -V \000"
```

```
        .align  2
$LC3:
        .ascii  "\011tp1 -i in_file -o out_file \000"
        .align  2
$LC4:
        .ascii  "Options: \012 \000"
        .align  2
$LC5:
        .ascii  "\011-V, --version     Print version and quit.\000"
        .align  2
$LC6:
        .ascii  "\011-h, --help        Print this information and quit. \000"
        .align  2
$LC7:
        .ascii  "\011-i, --input       Specify input stream/file, '-' for"
        .ascii  " stdin. \000"
        .align  2
$LC8:
        .ascii  "\011-o, --output      Specify output stream/file, '-' fo"
        .ascii  "r stdout \000"
        .align  2
$LC9:
        .ascii  "Examples: \000"
        .align  2
$LC10:
        .ascii  "\011tp1 < in.txt > out.txt \000"
        .align  2
$LC11:
        .ascii  "\011cat in.txt | tp1 -i - > out.txt \000"
        .text
        .align  2
        .globl  mensaje_ayuda
$LFB2 = .
        .file 1 "tp1.c"
        .loc 1 8 0
        .cfi_startproc
        .set    nomips16
        .set    nomicromips
        .ent    mensaje_ayuda
        .type    mensaje_ayuda, @function
mensaje_ayuda:
        .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
        .loc 1 9 0
        lw        $2,%got($LC0)($28)
        addiu        $4,$2,%lo($LC0)
        lw        $2,%call16(puts)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,puts
1:        jalr        $25
        nop

$LVL0 = .
        lw        $28,16($fp)
        .loc 1 10 0
        lw        $2,%got($LC1)($28)
        addiu        $4,$2,%lo($LC1)
        lw        $2,%call16(puts)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,puts
1:        jalr        $25
        nop

$LVL1 = .
        lw        $28,16($fp)
        .loc 1 11 0
        lw        $2,%got($LC2)($28)
        addiu        $4,$2,%lo($LC2)
        lw        $2,%call16(puts)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,puts
1:        jalr        $25
        nop

$LVL2 = .
        lw        $28,16($fp)
        .loc 1 12 0
        lw        $2,%got($LC3)($28)
        addiu        $4,$2,%lo($LC3)
        lw        $2,%call16(puts)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,puts
1:        jalr        $25
        nop

$LVL3 = .
        lw        $28,16($fp)
        .loc 1 13 0
        lw        $2,%got($LC4)($28)
        addiu        $4,$2,%lo($LC4)
        lw        $2,%call16(printf)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,printf
1:        jalr        $25
        nop
```

```
$LVL4 = .
        lw        $28,16($fp)
        .loc 1 14 0
        lw        $2,%got($LC5)($28)
        addiu        $4,$2,%lo($LC5)
        lw        $2,%call16(puts)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,puts
1:        jalr        $25
        nop

$LVL5 = .
        lw        $28,16($fp)
        .loc 1 15 0
        lw        $2,%got($LC6)($28)
        addiu        $4,$2,%lo($LC6)
        lw        $2,%call16(puts)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,puts
1:        jalr        $25
        nop

$LVL6 = .
        lw        $28,16($fp)
        .loc 1 16 0
        lw        $2,%got($LC7)($28)
        addiu        $4,$2,%lo($LC7)
        lw        $2,%call16(puts)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,puts
1:        jalr        $25
        nop

$LVL7 = .
        lw        $28,16($fp)
        .loc 1 17 0
        lw        $2,%got($LC8)($28)
        addiu        $4,$2,%lo($LC8)
        lw        $2,%call16(puts)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,puts
1:        jalr        $25
        nop

$LVL8 = .
        lw        $28,16($fp)
        .loc 1 18 0
        lw        $2,%got($LC9)($28)
        addiu        $4,$2,%lo($LC9)
        lw        $2,%call16(puts)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,puts
1:        jalr        $25
```

```
        nop

$LVL9 = .
        lw      $28,16($fp)
        .loc 1 19 0
        lw      $2,%got($LC10)($28)
        addiu      $4,$2,%lo($LC10)
        lw      $2,%call16(puts)($28)
        move      $25,$2
        .reloc      1f,R_MIPS_JALR,puts
1:      jalr      $25
        nop

$LVL10 = .
        lw      $28,16($fp)
        .loc 1 20 0
        lw      $2,%got($LC11)($28)
        addiu      $4,$2,%lo($LC11)
        lw      $2,%call16(puts)($28)
        move      $25,$2
        .reloc      1f,R_MIPS_JALR,puts
1:      jalr      $25
        nop

$LVL11 = .
        lw      $28,16($fp)
        .loc 1 21 0
        nop
        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      mensaje_ayuda
        .cfi_endproc
$LFE2:
        .size      mensaje_ayuda, .-mensaje_ayuda
        .rdata
        .align      2
$LC12:
        .ascii      "-h\000"
        .align      2
$LC13:
        .ascii      "-V\000"
        .align      2
$LC15:
```

```
        .ascii        "Versi\303\263n del TP: %.1f \012\000"
        .align        2
$LC16:
        .ascii        "-i\000"
        .align        2
$LC17:
        .ascii        "-\000"
        .align        2
$LC18:
        .ascii        "-o\000"
        .align        2
$LC19:
        .ascii        "r\000"
        .align        2
$LC20:
        .ascii        "w\000"
        .align        2
$LC21:
        .ascii        "%d %d\000"
        .align        2
$LC22:
        .ascii        "Memory Alloc Error\000"
        .align        2
$LC23:
        .ascii        "GCD(%d,%d) = %d \012\000"
        .text
        .align        2
        .globl        main
$LFB3 = .
        .loc 1 23 0
        .cfi_startproc
        .set        nomips16
        .set        nomicromips
        .ent        main
        .type        main, @function
main:
        .frame        $fp,144,$31               # vars= 104, regs= 2/0, args= 24, gp= 8
        .mask        0xc0000000,-4
        .fmask        0x00000000,0
        .set        noreorder
        .cpload        $25
        .set        nomacro
        addiu        $sp,$sp,-144
        .cfi_def_cfa_offset 144
        sw        $31,140($sp)
        sw        $fp,136($sp)
        .cfi_offset 31, -4
        .cfi_offset 30, -8
        move        $fp,$sp
        .cfi_def_cfa_register 30
        .cprestore        24
        sw        $4,144($fp)
        sw        $5,148($fp)
        .loc 1 26 0
```

```
        sw      $0,40($fp)
        .loc 1 27 0
        sw      $0,44($fp)
        .loc 1 31 0
        lw      $2,%got(stdin)($28)
        lw      $2,0($2)
        sw      $2,52($fp)
        .loc 1 32 0
        lw      $2,%got(stdout)($28)
        lw      $2,0($2)
        sw      $2,56($fp)
        .loc 1 34 0
        li      $2,1                    # 0x1
        sw      $2,48($fp)
        b       $L3
        nop

$L11:
        .loc 1 37 0
        lw      $2,48($fp)
        sll     $2,$2,2
        lw      $3,148($fp)
        addu    $2,$3,$2
        lw      $3,0($2)
        lw      $2,%got($LC12)($28)
        addiu   $5,$2,%lo($LC12)
        move    $4,$3
        lw      $2,%call16(strcmp)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,strcmp
1:      jalr    $25
        nop

$LVL12 = .
        lw      $28,24($fp)
        bne     $2,$0,$L4
        nop

        .loc 1 38 0
        lw      $2,%got(mensaje_ayuda)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,mensaje_ayuda
1:      jalr    $25
        nop

$LVL13 = .
        lw      $28,24($fp)
        .loc 1 39 0
        move    $2,$0
        b       $L26
        nop

$L4:
        .loc 1 42 0
```

```
        lw      $2,48($fp)
        sll     $2,$2,2
        lw      $3,148($fp)
        addu    $2,$3,$2
        lw      $3,0($2)
        lw      $2,%got($LC13)($28)
        addiu   $5,$2,%lo($LC13)
        move    $4,$3
        lw      $2,%call16(strcmp)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,strcmp
1:      jalr    $25
        nop

$LVL14 = .
        lw      $28,24($fp)
        bne     $2,$0,$L6
        nop

        .loc 1 43 0
        lw      $2,%got($LC14)($28)
        ldc1    $f0,%lo($LC14)($2)
        mfc1    $7,$f0
        mfhc1   $6,$f0
        lw      $2,%got($LC15)($28)
        addiu   $4,$2,%lo($LC15)
        lw      $2,%call16(printf)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,printf
1:      jalr    $25
        nop

$LVL15 = .
        lw      $28,24($fp)
        .loc 1 44 0
        move    $2,$0
        b       $L26
        nop

$L6:
        .loc 1 47 0
        lw      $2,48($fp)
        sll     $2,$2,2
        lw      $3,148($fp)
        addu    $2,$3,$2
        lw      $3,0($2)
        lw      $2,%got($LC16)($28)
        addiu   $5,$2,%lo($LC16)
        move    $4,$3
        lw      $2,%call16(strcmp)($28)
        move    $25,$2
        .reloc  1f,R_MIPS_JALR,strcmp
1:      jalr    $25
        nop
```

```
$LVL16 = .
        lw      $28,24($fp)
        bne      $2,$0,$L7
        nop

        .loc 1 49 0
        lw      $2,48($fp)
        addiu      $2,$2,1
        sll      $2,$2,2
        lw      $3,148($fp)
        addu      $2,$3,$2
        lw      $3,0($2)
        lw      $2,%got($LC17)($28)
        addiu      $5,$2,%lo($LC17)
        move      $4,$3
        lw      $2,%call16(strcmp)($28)
        move      $25,$2
        .reloc      1f,R_MIPS_JALR,strcmp
1:      jalr      $25
        nop

$LVL17 = .
        lw      $28,24($fp)
        bne      $2,$0,$L8
        nop

        .loc 1 50 0
        sw      $0,44($fp)
        b      $L7
        nop

$L8:
        .loc 1 53 0
        li      $2,1                    # 0x1
        sw      $2,44($fp)
        .loc 1 54 0
        lw      $2,48($fp)
        addiu      $2,$2,1
        sll      $2,$2,2
        lw      $3,148($fp)
        addu      $2,$3,$2
        lw      $2,0($2)
        sw      $2,32($fp)
$L7:
        .loc 1 58 0
        lw      $2,48($fp)
        sll      $2,$2,2
        lw      $3,148($fp)
        addu      $2,$3,$2
        lw      $3,0($2)
        lw      $2,%got($LC18)($28)
        addiu      $5,$2,%lo($LC18)
        move      $4,$3
```

```
        lw      $2,%call16(strcmp)($28)
        move      $25,$2
        .reloc      1f,R_MIPS_JALR,strcmp
1:        jalr      $25
        nop

$LVL18 = .
        lw      $28,24($fp)
        bne       $2,$0,$L9
        nop

        .loc 1 60 0
        lw      $2,48($fp)
        addiu       $2,$2,1
        sll       $2,$2,2
        lw      $3,148($fp)
        addu      $2,$3,$2
        lw      $3,0($2)
        lw      $2,%got($LC17)($28)
        addiu       $5,$2,%lo($LC17)
        move      $4,$3
        lw      $2,%call16(strcmp)($28)
        move      $25,$2
        .reloc      1f,R_MIPS_JALR,strcmp
1:        jalr      $25
        nop

$LVL19 = .
        lw      $28,24($fp)
        bne       $2,$0,$L10
        nop

        .loc 1 61 0
        sw      $0,40($fp)
        b       $L9
        nop

$L10:
        .loc 1 64 0
        li      $2,1                    # 0x1
        sw      $2,40($fp)
        .loc 1 65 0
        lw      $2,48($fp)
        addiu       $2,$2,1
        sll       $2,$2,2
        lw      $3,148($fp)
        addu      $2,$3,$2
        lw      $2,0($2)
        sw      $2,36($fp)
$L9:
        .loc 1 34 0 discriminator 2
        lw      $2,48($fp)
        addiu       $2,$2,1
        sw      $2,48($fp)
```

```
$L3:
        .loc 1 34 0 is_stmt 0 discriminator 1
        lw        $3,48($fp)
        lw        $2,144($fp)
        slt        $2,$3,$2
        bne        $2,$0,$L11
        nop

        .loc 1 71 0 is_stmt 1
        lw        $3,44($fp)
        li        $2,1                        # 0x1
        bne        $3,$2,$L12
        nop

        .loc 1 72 0
        lw        $2,%got($LC19)($28)
        addiu        $5,$2,%lo($LC19)
        lw        $4,32($fp)
        lw        $2,%call16(fopen)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,fopen
1:        jalr        $25
        nop

$LVL20 = .
        lw        $28,24($fp)
        sw        $2,52($fp)
        .loc 1 74 0
        lw        $2,52($fp)
        bne        $2,$0,$L12
        nop

        .loc 1 75 0
        li        $2,-1                         # 0xffffffffffffffff
        b        $L26
        nop

$L12:
        .loc 1 79 0
        lw        $3,40($fp)
        li        $2,1                        # 0x1
        bne        $3,$2,$L13
        nop

        .loc 1 80 0
        lw        $2,%got($LC20)($28)
        addiu        $5,$2,%lo($LC20)
        lw        $4,36($fp)
        lw        $2,%call16(fopen)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,fopen
1:        jalr        $25
        nop
```

```
$LVL21 = .
        lw        $28,24($fp)
        sw        $2,56($fp)
        .loc 1 82 0
        lw        $2,56($fp)
        bne        $2,$0,$L13
        nop

        .loc 1 83 0
        li        $2,-1                         # 0xffffffffffffffff
        b        $L26
        nop

$L13:
        .loc 1 87 0
        sw        $0,60($fp)
        .loc 1 88 0
        sw        $0,64($fp)
        .loc 1 92 0
        lw        $4,52($fp)
        lw        $2,%call16(feof)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,feof
1:        jalr        $25
        nop

$LVL22 = .
        lw        $28,24($fp)
        beq        $2,$0,$L14
        nop

        .loc 1 93 0
        move        $2,$0
        b        $L26
        nop

$L14:
        .loc 1 96 0
        addiu        $2,$fp,72
        lw        $6,52($fp)
        li        $5,50                         # 0x32
        move        $4,$2
        lw        $2,%call16(fgets)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,fgets
1:        jalr        $25
        nop

$LVL23 = .
        lw        $28,24($fp)
        .loc 1 97 0
        b        $L15
        nop
```

```
$L17:
$LBB2 = .
        .loc 1 100 0
        addiu        $2,$fp,124
        addiu        $4,$2,4
        addiu        $2,$fp,124
        addiu        $3,$fp,72
        move        $7,$4
        move        $6,$2
        lw        $2,%got($LC21)($28)
        addiu        $5,$2,%lo($LC21)
        move        $4,$3
        lw        $2,%call16(__isoc99_sscanf)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,__isoc99_sscanf
1:        jalr        $25
        nop

$LVL24 = .
        lw        $28,24($fp)
        .loc 1 102 0
        lw        $2,64($fp)
        addiu        $3,$2,1
        move        $2,$3
        sll        $2,$2,1
        addu        $2,$2,$3
        sll        $2,$2,2
        move        $5,$2
        lw        $4,60($fp)
        lw        $2,%call16(realloc)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,realloc
1:        jalr        $25
        nop

$LVL25 = .
        lw        $28,24($fp)
        sw        $2,68($fp)
        .loc 1 103 0
        lw        $2,68($fp)
        bne        $2,$0,$L16
        nop

        .loc 1 104 0
        lw        $2,%got($LC22)($28)
        addiu        $4,$2,%lo($LC22)
        lw        $2,%call16(puts)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,puts
1:        jalr        $25
        nop

$LVL26 = .
        lw        $28,24($fp)
```

```
        .loc 1 105 0
        li        $2,-1                         # 0xffffffffffffffff
        b        $L26
        nop

$L16:
        .loc 1 107 0
        lw        $2,68($fp)
        sw        $2,60($fp)
        .loc 1 108 0
        lw        $3,64($fp)
        move        $2,$3
        sll        $2,$2,1
        addu        $2,$2,$3
        sll        $2,$2,2
        move        $3,$2
        lw        $2,60($fp)
        addu        $2,$2,$3
        lw        $5,124($fp)
        lw        $4,128($fp)
        lw        $3,132($fp)
        sw        $5,0($2)
        sw        $4,4($2)
        sw        $3,8($2)
        .loc 1 109 0
        lw        $2,64($fp)
        addiu        $2,$2,1
        sw        $2,64($fp)
        .loc 1 111 0
        addiu        $2,$fp,72
        lw        $6,52($fp)
        li        $5,50                         # 0x32
        move        $4,$2
        lw        $2,%call16(fgets)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,fgets
1:        jalr        $25
        nop

$LVL27 = .
        lw        $28,24($fp)
$L15:
$LBE2 = .
        .loc 1 97 0
        lw        $4,52($fp)
        lw        $2,%call16(feof)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,feof
1:        jalr        $25
        nop

$LVL28 = .
        lw        $28,24($fp)
        beq        $2,$0,$L17
```

```
        nop

        .loc 1 115 0
        lw        $2,64($fp)
        bne        $2,$0,$L18
        nop

        .loc 1 116 0
        lw        $3,44($fp)
        li        $2,1                        # 0x1
        bne        $3,$2,$L19
        nop

        .loc 1 117 0
        lw        $4,52($fp)
        lw        $2,%call16(fclose)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,fclose
1:        jalr        $25
        nop

$LVL29 = .
        lw        $28,24($fp)
$L19:
        .loc 1 119 0
        lw        $3,40($fp)
        li        $2,1                        # 0x1
        bne        $3,$2,$L20
        nop

        .loc 1 120 0
        lw        $4,56($fp)
        lw        $2,%call16(fclose)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,fclose
1:        jalr        $25
        nop

$LVL30 = .
        lw        $28,24($fp)
$L20:
        .loc 1 122 0
        li        $2,-1                        # 0xffffffffffffffff
        b        $L26
        nop

$L18:
        .loc 1 126 0
        lw        $5,64($fp)
        lw        $4,60($fp)
        lw        $2,%call16(euclides)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,euclides
1:        jalr        $25
```

```
        nop

$LVL31 = .
        lw        $28,24($fp)
        .loc 1 129 0
        sw        $0,48($fp)
        b        $L21
        nop

$L22:
        .loc 1 130 0 discriminator 3
        lw        $3,48($fp)
        move        $2,$3
        sll        $2,$2,1
        addu        $2,$2,$3
        sll        $2,$2,2
        move        $3,$2
        lw        $2,60($fp)
        addu        $2,$2,$3
        lw        $4,0($2)
        lw        $3,48($fp)
        move        $2,$3
        sll        $2,$2,1
        addu        $2,$2,$3
        sll        $2,$2,2
        move        $3,$2
        lw        $2,60($fp)
        addu        $2,$2,$3
        lw        $5,4($2)
        lw        $3,48($fp)
        move        $2,$3
        sll        $2,$2,1
        addu        $2,$2,$3
        sll        $2,$2,2
        move        $3,$2
        lw        $2,60($fp)
        addu        $2,$2,$3
        lw        $2,8($2)
        sw        $2,16($sp)
        move        $7,$5
        move        $6,$4
        lw        $2,%got($LC23)($28)
        addiu        $5,$2,%lo($LC23)
        lw        $4,56($fp)
        lw        $2,%call16(fprintf)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,fprintf
1:        jalr        $25
        nop

$LVL32 = .
        lw        $28,24($fp)
        .loc 1 129 0 discriminator 3
        lw        $2,48($fp)
```

```
        addiu        $2,$2,1
        sw        $2,48($fp)
$L21:
        .loc 1 129 0 is_stmt 0 discriminator 1
        lw        $3,48($fp)
        lw        $2,64($fp)
        sltu        $2,$3,$2
        bne        $2,$0,$L22
        nop

        .loc 1 133 0 is_stmt 1
        lw        $3,44($fp)
        li        $2,1                        # 0x1
        bne        $3,$2,$L23
        nop

        .loc 1 134 0
        lw        $4,52($fp)
        lw        $2,%call16(fclose)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,fclose
1:        jalr        $25
        nop

$LVL33 = .
        lw        $28,24($fp)
$L23:
        .loc 1 136 0
        lw        $3,40($fp)
        li        $2,1                        # 0x1
        bne        $3,$2,$L24
        nop

        .loc 1 137 0
        lw        $4,56($fp)
        lw        $2,%call16(fclose)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,fclose
1:        jalr        $25
        nop

$LVL34 = .
        lw        $28,24($fp)
$L24:
        .loc 1 139 0
        lw        $2,60($fp)
        beq        $2,$0,$L25
        nop

        .loc 1 140 0
        lw        $4,60($fp)
        lw        $2,%call16(free)($28)
        move        $25,$2
        .reloc        1f,R_MIPS_JALR,free
```

```
1:        jalr        $25
        nop

$LVL35 = .
        lw        $28,24($fp)
$L25:
        .loc 1 142 0
        move        $2,$0
$L26:
        .loc 1 143 0 discriminator 1
        move        $sp,$fp
        .cfi_def_cfa_register 29
        lw        $31,140($sp)
        lw        $fp,136($sp)
        addiu        $sp,$sp,144
        .cfi_restore 30
        .cfi_restore 31
        .cfi_def_cfa_offset 0
        jr        $31
        nop

        .set        macro
        .set        reorder
        .end        main
        .cfi_endproc
$LFE3:
        .size        main, .-main
        .rdata
        .align        3
$LC14:
        .word        1072902963
        .word        858993459
        .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 "gcd.h"
        .section        .debug_info,"",@progbits
$Ldebug_info0:
        .4byte        0x476
        .2byte        0x4
        .4byte        $Ldebug_abbrev0
        .byte        0x4
        .uleb128 0x1
        .4byte        $LASF73
        .byte        0xc
        .4byte        $LASF74
        .4byte        $LASF75
        .4byte        $Ltext0
        .4byte        $Letext0-$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 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
.4byte        0xa0
.byte        0x2c
.uleb128 0xa
.4byte        $LASF28
.byte        0x5
.2byte        0x104
```

```
.4byte        0x272
.byte        0x30
.uleb128 0xa
.4byte        $LASF29
.byte        0x5
.2byte        0x106
.4byte        0x278
.byte        0x34
.uleb128 0xa
.4byte        $LASF30
.byte        0x5
.2byte        0x108
.4byte        0x5a
.byte        0x38
.uleb128 0xa
.4byte        $LASF31
.byte        0x5
.2byte        0x10c
.4byte        0x5a
.byte        0x3c
.uleb128 0xa
.4byte        $LASF32
.byte        0x5
.2byte        0x10e
.4byte        0x7a
.byte        0x40
.uleb128 0xa
.4byte        $LASF33
.byte        0x5
.2byte        0x112
.4byte        0x3e
.byte        0x44
.uleb128 0xa
.4byte        $LASF34
.byte        0x5
.2byte        0x113
.4byte        0x4c
.byte        0x46
.uleb128 0xa
.4byte        $LASF35
.byte        0x5
.2byte        0x114
.4byte        0x27e
.byte        0x47
.uleb128 0xa
.4byte        $LASF36
.byte        0x5
.2byte        0x118
.4byte        0x28e
.byte        0x48
.uleb128 0xa
.4byte        $LASF37
.byte        0x5
.2byte        0x121
```

```
.4byte        0x8c
.byte        0x50
.uleb128 0xa
.4byte        $LASF38
.byte        0x5
.2byte        0x129
.4byte        0x9e
.byte        0x58
.uleb128 0xa
.4byte        $LASF39
.byte        0x5
.2byte        0x12a
.4byte        0x9e
.byte        0x5c
.uleb128 0xa
.4byte        $LASF40
.byte        0x5
.2byte        0x12b
.4byte        0x9e
.byte        0x60
.uleb128 0xa
.4byte        $LASF41
.byte        0x5
.2byte        0x12c
.4byte        0x9e
.byte        0x64
.uleb128 0xa
.4byte        $LASF42
.byte        0x5
.2byte        0x12e
.4byte        0x25
.byte        0x68
.uleb128 0xa
.4byte        $LASF43
.byte        0x5
.2byte        0x12f
.4byte        0x5a
.byte        0x6c
.uleb128 0xa
.4byte        $LASF44
.byte        0x5
.2byte        0x131
.4byte        0x294
.byte        0x70
.byte        0
.uleb128 0xb
.4byte        $LASF76
.byte        0x5
.byte        0x96
.uleb128 0x8
.4byte        $LASF46
.byte        0xc
.byte        0x5
.byte        0x9c
```

```
.4byte        0x272
.uleb128 0x9
.4byte        $LASF47
.byte        0x5
.byte        0x9d
.4byte        0x272
.byte        0
.uleb128 0x9
.4byte        $LASF48
.byte        0x5
.byte        0x9e
.4byte        0x278
.byte        0x4
.uleb128 0x9
.4byte        $LASF49
.byte        0x5
.byte        0xa2
.4byte        0x5a
.byte        0x8
.byte        0
.uleb128 0x6
.byte        0x4
.4byte        0x241
.uleb128 0x6
.byte        0x4
.4byte        0xbd
.uleb128 0xc
.4byte        0xa6
.4byte        0x28e
.uleb128 0xd
.4byte        0x97
.byte        0
.byte        0
.uleb128 0x6
.byte        0x4
.4byte        0x23a
.uleb128 0xc
.4byte        0xa6
.4byte        0x2a4
.uleb128 0xd
.4byte        0x97
.byte        0x27
.byte        0
.uleb128 0xe
.4byte        $LASF77
.uleb128 0xf
.4byte        $LASF50
.byte        0x5
.2byte        0x13b
.4byte        0x2a4
.uleb128 0xf
.4byte        $LASF51
.byte        0x5
.2byte        0x13c
```

```
.4byte        0x2a4
.uleb128 0xf
.4byte        $LASF52
.byte        0x5
.2byte        0x13d
.4byte        0x2a4
.uleb128 0x6
.byte        0x4
.4byte        0xad
.uleb128 0x7
.4byte        0x2cd
.uleb128 0x10
.4byte        $LASF53
.byte        0x4
.byte        0xaa
.4byte        0x278
.uleb128 0x10
.4byte        $LASF54
.byte        0x4
.byte        0xab
.4byte        0x278
.uleb128 0x10
.4byte        $LASF55
.byte        0x4
.byte        0xac
.4byte        0x278
.uleb128 0x10
.4byte        $LASF56
.byte        0x6
.byte        0x1a
.4byte        0x5a
.uleb128 0xc
.4byte        0x2d3
.4byte        0x30f
.uleb128 0x11
.byte        0
.uleb128 0x7
.4byte        0x304
.uleb128 0x10
.4byte        $LASF57
.byte        0x6
.byte        0x1b
.4byte        0x30f
.uleb128 0x3
.byte        0x8
.byte        0x4
.4byte        $LASF58
.uleb128 0x12
.ascii        "gcd\000"
.byte        0xc
.byte        0x7
.byte        0x6
.4byte        0x357
.uleb128 0x9
```

```
.4byte        $LASF59
.byte        0x7
.byte        0x7
.4byte        0x5a
.byte        0
.uleb128 0x9
.4byte        $LASF60
.byte        0x7
.byte        0x8
.4byte        0x5a
.byte        0x4
.uleb128 0x9
.4byte        $LASF61
.byte        0x7
.byte        0x9
.4byte        0x5a
.byte        0x8
.byte        0
.uleb128 0x13
.4byte        $LASF78
.byte        0x1
.byte        0x17
.4byte        0x5a
.4byte        $LFB3
.4byte        $LFE3-$LFB3
.uleb128 0x1
.byte        0x9c
.4byte        0x446
.uleb128 0x14
.4byte        $LASF62
.byte        0x1
.byte        0x17
.4byte        0x5a
.uleb128 0x2
.byte        0x91
.sleb128 0
.uleb128 0x14
.4byte        $LASF63
.byte        0x1
.byte        0x17
.4byte        0x446
.uleb128 0x2
.byte        0x91
.sleb128 4
.uleb128 0x15
.4byte        $LASF64
.byte        0x1
.byte        0x18
.4byte        0xa0
.uleb128 0x3
.byte        0x91
.sleb128 -112
.uleb128 0x15
.4byte        $LASF65
```

```
.byte        0x1
.byte        0x19
.4byte        0xa0
.uleb128 0x3
.byte        0x91
.sleb128 -108
.uleb128 0x15
.4byte        $LASF66
.byte        0x1
.byte        0x1a
.4byte        0x5a
.uleb128 0x3
.byte        0x91
.sleb128 -104
.uleb128 0x15
.4byte        $LASF67
.byte        0x1
.byte        0x1b
.4byte        0x5a
.uleb128 0x3
.byte        0x91
.sleb128 -100
.uleb128 0x16
.ascii        "i\000"
.byte        0x1
.byte        0x1d
.4byte        0x5a
.uleb128 0x3
.byte        0x91
.sleb128 -96
.uleb128 0x15
.4byte        $LASF68
.byte        0x1
.byte        0x1f
.4byte        0x44c
.uleb128 0x3
.byte        0x91
.sleb128 -92
.uleb128 0x15
.4byte        $LASF69
.byte        0x1
.byte        0x20
.4byte        0x44c
.uleb128 0x3
.byte        0x91
.sleb128 -88
.uleb128 0x15
.4byte        $LASF70
.byte        0x1
.byte        0x57
.4byte        0x452
.uleb128 0x3
.byte        0x91
.sleb128 -84
```

```
.uleb128 0x15
.4byte        $LASF71
.byte        0x1
.byte        0x58
.4byte        0x25
.uleb128 0x3
.byte        0x91
.sleb128 -80
.uleb128 0x15
.4byte        $LASF72
.byte        0x1
.byte        0x5a
.4byte        0x458
.uleb128 0x3
.byte        0x91
.sleb128 -72
.uleb128 0x17
.4byte        $LBB2
.4byte        $LBE2-$LBB2
.uleb128 0x16
.ascii        "p\000"
.byte        0x1
.byte        0x62
.4byte        0x326
.uleb128 0x2
.byte        0x91
.sleb128 -20
.uleb128 0x16
.ascii        "try\000"
.byte        0x1
.byte        0x66
.4byte        0x452
.uleb128 0x3
.byte        0x91
.sleb128 -76
.byte        0
.byte        0
.uleb128 0x6
.byte        0x4
.4byte        0xa0
.uleb128 0x6
.byte        0x4
.4byte        0xb2
.uleb128 0x6
.byte        0x4
.4byte        0x326
.uleb128 0xc
.4byte        0xa6
.4byte        0x468
.uleb128 0xd
.4byte        0x97
.byte        0x31
.byte        0
.uleb128 0x18
```

```
        .4byte        $LASF79
        .byte        0x1
        .byte        0x8
        .4byte        $LFB2
        .4byte        $LFE2-$LFB2
        .uleb128 0x1
        .byte        0x9c
        .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 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 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 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 0x17
.uleb128 0xb
.byte        0x1
.uleb128 0x11
.uleb128 0x1
.uleb128 0x12
.uleb128 0x6
.byte        0
.byte        0
.uleb128 0x18
.uleb128 0x2e
.byte        0
.uleb128 0x3f
.uleb128 0x19
.uleb128 0x3
.uleb128 0xe
.uleb128 0x3a
.uleb128 0xb
.uleb128 0x3b
.uleb128 0xb
.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       $Letext0-$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"
$LASF30:
        .ascii       "_fileno\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"
$LASF59:
        .ascii       "num_a\000"
```

```
$LASF32:
        .ascii        "_old_offset\000"
$LASF37:
        .ascii        "_offset\000"
$LASF46:
        .ascii        "_IO_marker\000"
$LASF53:
        .ascii        "stdin\000"
$LASF0:
        .ascii        "unsigned int\000"
$LASF3:
        .ascii        "long unsigned int\000"
$LASF70:
        .ascii        "puntero\000"
$LASF77:
        .ascii        "_IO_FILE_plus\000"
$LASF21:
        .ascii        "_IO_write_ptr\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"
$LASF50:
        .ascii        "_IO_2_1_stdin_\000"
$LASF65:
        .ascii        "output\000"
$LASF73:
        .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"
$LASF58:
        .ascii        "long double\000"
$LASF68:
        .ascii        "file_in\000"
$LASF22:
        .ascii        "_IO_write_end\000"
$LASF76:
        .ascii        "_IO_lock_t\000"
$LASF45:
        .ascii        "_IO_FILE\000"
$LASF49:
```

```
        .ascii  "_pos\000"
$LASF57:
        .ascii  "sys_errlist\000"
$LASF28:
        .ascii  "_markers\000"
$LASF66:
        .ascii  "modo_out\000"
$LASF1:
        .ascii  "unsigned char\000"
$LASF5:
        .ascii  "short int\000"
$LASF67:
        .ascii  "modo_in\000"
$LASF34:
        .ascii  "_vtable_offset\000"
$LASF60:
        .ascii  "num_b\000"
$LASF51:
        .ascii  "_IO_2_1_stdout_\000"
$LASF15:
        .ascii  "FILE\000"
$LASF64:
        .ascii  "input\000"
$LASF74:
        .ascii  "tp1.c\000"
$LASF69:
        .ascii  "file_out\000"
$LASF61:
        .ascii  "gcd_ab\000"
$LASF75:
        .ascii  "/root/tmp/tp\000"
$LASF14:
        .ascii  "char\000"
$LASF71:
        .ascii  "largo_array\000"
$LASF72:
        .ascii  "buffer\000"
$LASF47:
        .ascii  "_next\000"
$LASF12:
        .ascii  "__off64_t\000"
$LASF19:
        .ascii  "_IO_read_base\000"
$LASF27:
        .ascii  "_IO_save_end\000"
$LASF79:
        .ascii  "mensaje_ayuda\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"
$LASF63:
        .ascii        "argv\000"
$LASF26:
        .ascii        "_IO_backup_base\000"
$LASF62:
        .ascii        "argc\000"
$LASF78:
        .ascii        "main\000"
$LASF20:
        .ascii        "_IO_write_base\000"
        .ident        "GCC: (Debian 6.3.0-18+deb9u1) 6.3.0 20170516"
```

4. Referencias

Enunciado del TP

- https://drive.google.com/file/d/1ET3eRq6Qs6Yh5KYEQNRmkeWweio3g1NI/view?usp=sharing