Facultad de Ingeniería
Universidad de Buenos Aires
Organizacion de Computadoras (66.20)

Trabajo Práctico N°0

2^{do} Cuatrimestre, 2019

Boada, Ignacio Daniel	95212	ignacio.boada@outlook.com
Goñi, Mauro Ariel	87646	maurogoni@gmail.com
Perez Machado, Axel Mauro	101127	axelmpm@gmail.com

1. Comandos para compilación

Los comandos que ejecutamos para compilar el programa en Linux y NETBSD respectivamente fueron exactamente los provistos por la catedra sin ninguna modificación, si con la el recado de ejecutarlos sobre el directorio contenedor del archivo C.

Estos son para LINUX:
gcc -Wall -o Tp0 Tp0.c

Para NETBSD:
gcc -Wall -O0 Tp0.cmediumskip
gcc -Wall -O0 -S -mrnames Tp0.c (en el caso de querer generar solo el archivo assembly para MIPS32.

2. Diseño e implementación

Nuestra implementación no es nada fuera de lo común y busca ser lo mas simple y directa posible.

1. Primero se leen todos lo inputs en la primer linea de stdin y se almacenan en un array
2. Se crean las dos matrices
3. Se llenan sus elementos con los valores leidos
4. Se realiza el producto que luego se almacena en una tercera matriz que se crea
5. Se envia el resultado por stdout
6. Se libera memoria
7. Se vuelve a empezar todo el proceso con la siguiente linea de stdin

8. Se crea un punto de acceso global sobre la variable que almacena todo el input de una linea para que, en caso de surgir un error, la funcion que maneja el error pueda siempre liberar memoria.

Consideraciones:

1. Si algun error de formato aparece, este se valida en el primer paso, se informa del problema por stderr y se corta el programa
2. Para facilitar la implementación no consideramos un error de formato una dimension positiva no nula de tipo float o double pero de mantiza nula .
3. Para facilitar la implementación no consideramos como un error de formato un archivo vacio. Ante tal situacion simplemente no se hace nada y se cierra el programa
4. Ante el ingreso de comandos -h,-V,–help,–version, estos se manejan abriendo un archivo de texto que se encuentra en el mismo directorio que el archivo compilado y el fuente. Esto permite cambiar el mensaje a mostrar sin tocar el codigo ni recompilar.

3. Pruebas

Para una completa descripción de las pruebas corridas se tiene la carpeta entregada de "Pruebas"

En ella se encuentra una bateria de 27 pruebas que evaluan a grandes rasgos:

1. Errores de dimension
2. Errores de formato en elementos
3. Errores en alguna linea del archivo con otras en formato correcto
4. Producto entre identidad y una matriz cualquiera
5. Producto entre matriz nula y una matriz cualquiera
6. Producto entre una matriz inversible y su inversa
7. Varios productos consecutivos sin errores
8. Productos usando matrices grandes y chicas

Entre estas tenemos como ejemplos:

(Prueba 27)
2 1 2 3 4 1 2 3 4 3 1 2 3 4 5 6.1 3 2 1 1 0 0 0 1 0 0 0 1

Ejemplo provisto por la cátedra.

(Prueba 4)
2 4 8 9 7 1 0 0 1

Ejemplo de producto por identidad.

(Prueba 13)
2.32 8 4 6 5 8 7 7 8

Ejemplo de caso con dimensión no entera.

(Prueba 25)
2 4 5 7 2 -0.074074074 0.185185185 0.259259259 -0.148148148

Ejemplo bastante interesante donde se prueban elementos con mantiza y que ademas son negativos y donde ademas se testea el aspecto numérico del programa, es decir, la precisión.

4. Conclusiones

El trabajo práctico realizado fue introductorio para que comenzemos a estar relacionarnos con las herramientas que se utilizan en la cátedra.

Considerando la finalidad del trabajo práctico podemos concluir que fue de gran utilidad, ya que, en primer lugar tuvimos que aprender la sintaxis de C y comprender cómo este lenguaje maneja archivos y memoria. Otro punto importante fue que tuvimos que utilizar por primera vez el entorno NetBSD para correr el programa y obtener el código en MIPS, lo que será útil para el siguiente trabajo práctico ya que se utilizará el lenguaje que implementa MIPS.

A lo largo del tp nos encontramos con muchos errores que se debieron solucionar, propios de una mala utilización del lenguaje C, por lo que tuvimos que mejorar el uso de funciones y la modularización de las mismas. Con este trabajo tuvimos que considerar cada uno de los posibles errores que arrojan las librerías de C y pensar cómo debería reaccionar el programa ante cada uno de ellos. El hecho de tener que liberar memoria nos planteó problemas ya que teníamos que considerar mantener referencias siempre a las posiciones de memoria dinámica, y tenerlo en cuenta de liberarlas ante cada posible error. Una vez finalizada la programación y comenzadas las pruebas, nos dimos cuenta que habían muchas posibilidades de ingreso que no habíamos tenído en cuenta, por lo que debimos agregar validaciones y cortes del programa ante ingresos de datos inválidos.

Un punto no menor es que el trabajo se hizo en equipo

Una consideración que no es menor, es el trabajo fue en equipo. La comunicación en el equipo tampoco es sencilla ya que todos manejamos horarios diferentes y esto agrega una difucultad extra al problema de fondo, El aprendizaje que nos dejó este proceso es que es mejor organizarse bien desde un principio, y dividir tareas de manera bien modularizadas.

Finalmente podemos concluir que el trabajo nos dejó aprendizajes muy importantes para seguir con la materia, ya que aprender C es de suma utilidad, correr programas en NetBSD parece que se utilizará a lo largo de todo el curso, y sobretodo cómo trabajar en varias plataformas como Linux y NetBSD sin que hayan conflictos en la codificación permitiendo la portabilidad.

5. Código fuente C

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

double* input = NULL; //GLOBAL ACCESS VARIABLE

typedef struct matrix {
    size_t rows;
    size_t cols;
    double* array;
} matrix_t;


void freeInputArray(){
    if (input != NULL){
        free(input);
    }
    input = NULL;
}

void printArray(int len,double* array){
    int i;
    for(i=0;i<len;i++){
        printf("elemento %d: %g\n",i,array[i]);
    }
}

void raiseError(const char* s){

    fprintf(stderr,"\n");
    fprintf(stderr,"==========\n");
    fprintf(stderr,"ERROR MESSAGE: %s\n",s);
    fprintf(stderr,"==========\n");
    fprintf(stderr,"\n");

    freeInputArray();
    exit (EXIT_FAILURE);
}

char *readLine(FILE* fp){
//The size is extended by the input with the value of the provisional
    int size = 10; //HARDCODED
    char *str;
    int ch;
    size_t len = 0;

    str = realloc(NULL, sizeof(char)*size);//size is start size
    if(!str)return str;
    while(EOF!=(ch=fgetc(fp)) && ch != '\n'){
        str[len++]=ch;
        if(len==size){
            str = realloc(str, sizeof(char)*(size+=16)); //HARDCODED
            if(!str)return str;
```

```
        }
    }

    if (ferror(stdin) != 0){
        free(str);
        raiseError("FGETC ERROR: I/O error");
    }

    str[len++]='\0';

    str = realloc(str, sizeof(char)*len);

    return str;
}

void readElementsInLine(int dimention, double* array){

    char* line = readLine(stdin);
    char* head_line_pointer = line;

    float x;
    int offset;
    int i = 0;
    int returnValue;
    int cantidadDeElementosLeidos;

    while (true)
    {
        returnValue = sscanf(head_line_pointer, "%g%n", &x, &offset);
        if (ferror(stdin) != 0){
            free(array);
            free(line);
            raiseError("SSCANF ERROR: I/O error");
        }


        if (returnValue == 1){
            head_line_pointer += offset;
            array[i] = (double)x;
            continue;
            i++;
        }

        if (returnValue == -1){
            cantidadDeElementosLeidos = i;
            if(cantidadDeElementosLeidos != dimention*dimention*2){
                free(array);
                free(line);
                raiseError("No coincide dimension con cantidad de elementos ingresados");
            }
            break;
        }

        if (returnValue != 1){
            free(array);
            free(line);
```

```
            raiseError("Input no numerico");
            break;
        }

    }
}

double* readInput(int* dimention){

    float firstInputElement;//initialized as double to check if corrupted input
    double* array;
    int returnValue;

    //READ FIRST
    returnValue = fscanf(stdin,"%g", &firstInputElement);

    //CHECK IF END OF LINE
    if (returnValue == -1){
        if (ferror(stdin) != 0){{raiseError("FSCANF ERROR: I/O error");}}
        else{exit (0);} // en este caso se identifica que el EOF no se debe a un error y s
    }
    //CHECK IF INPUT IS NUMERIC
    if (returnValue != 1){
        raiseError("Dimension no numerica");
    }

    //CHECK IF INPUT IS TYPE UINT
    float mantiza = firstInputElement - (int)firstInputElement;
    if (mantiza > 0 || (firstInputElement <= 0)){
        raiseError("La dimension no es entera positiva");
    }

    //ALLOCATE MEMORY FOR MATRICES INPUT ELEMENTS
    (*dimention) = (int)firstInputElement;
    array = malloc(sizeof(double)*(*dimention)*(*dimention)*2);

    //CHECK IF ALLOCATION IS SUCCESSFULL
    if (array == NULL){
        raiseError("No se pudo allocar memoria para inputs");
    }
    //READ WHOLE LINE
    readElementsInLine((*dimention), array);

    return array;
}

void outputFile(FILE* out, char fileName[]){
    //ADAPTS FILE NAME
    char s[100] = "";
    strcat(s, "./");
    strcat(s, fileName);
    int return_value;

    //TRIES TO OPEN FILE
    FILE* fp;
    fp = fopen(s,"r");
```

```
    if(fp == NULL){
        raiseError("no se pudo abrir archivo de salida");
    }

    //OUTPUTS
    char c;
        while(c != EOF){
        c = getc(fp);
        if ((return_value = fprintf(out,"%c",c)) < 0){raiseError("FPRINTF ERROR: I/O error
        }

    if (ferror(stdin) != 0){
        raiseError("FGETC ERROR: I/O error");
    }
}

matrix_t* create_matrix(size_t rows, size_t cols){
    matrix_t *matriz = malloc(sizeof(matrix_t));
    if (matriz == NULL){ //si no puede reservar la memoria, deja el puntero en NULL
        raiseError("no se pudo allocar memoria para matriz");
    }
    matriz->array = malloc(sizeof(double) * cols * rows); //representara los elementos de
    if (matriz->array == NULL){ //si no puede reservar la memoria, deja el puntero en NULL
        free(matriz);
        raiseError("no se pudo allocar memoria para elementos de matriz");
    }
    matriz->rows = rows;
    matriz->cols = cols;
    return matriz;
}

void destroy_matrix(matrix_t* m){
    if (m != NULL){
        free(m->array);
        free(m);
    }
}

void fillUpMatrices(matrix_t* matrix_a, matrix_t* matrix_b, int dimention,double* input){

    int i;
    for (i = 0; i < dimention*dimention; i++){
        matrix_a->array[i] = input[i];
    }

    for (i = dimention*dimention; i < dimention*dimention*2; i++){
        matrix_b->array[i - dimention*dimention] = input[i];
    }
}

matrix_t* matrix_multiply(matrix_t* matrix_a,matrix_t* matrix_b){

    int dimention = matrix_a->rows;

    matrix_t* matrix_c = create_matrix(dimention,dimention);
```

```
    int row;
    int column;
    int i;
    int j;
    double element;

    for (i = 0; i < dimention*dimention; i++){

        row = (int)(i / dimention);
        column = (int)(i % dimention);

        element = 0;
        for (j = 0; j < dimention; j++){

            element += matrix_a->array[row*dimention + j] * matrix_b->array[j*dimention +
        }
        matrix_c->array[i] = element;
    }
    return matrix_c;
}

void print_matrix(FILE* out, matrix_t* matrix_m){

    int dimention = matrix_m->rows;
    double x;
    int i;
    int return_value;

    if ((return_value = fprintf(out,"%d",dimention)) < 0){raiseError("FPRINTF ERROR: I/O e
    if ((return_value = fprintf(out,"%c",' ')) < 0){raiseError("FPRINTF ERROR: I/O error")

        for (i = 0; i < dimention*dimention; i++){
        x = matrix_m->array[i];
        if ((return_value = fprintf(out,"%g",x)) < 0){raiseError("FPRINTF ERROR: I/O error
        if ((return_value = fprintf(out,"%c",' ')) < 0){raiseError("FPRINTF ERROR: I/O err
        }
    if ((return_value = fprintf(out,"\n")) < 0){raiseError("FPRINTF ERROR: I/O error");}
}

int main(int argc, const char* argv[]){

    //INITIALIZATION
    FILE* OUT = stdout;
    bool endProgram = false;

    //HANDELING COMANDS
    if (argc > 1){
        if (strcmp(argv[1],"-h") == 0 || strcmp(argv[1],"--help") == 0){
            char fileName[] = "help";
            outputFile(OUT, fileName);
            endProgram = true;
        }

        else if (strcmp(argv[1],"-V") == 0 || strcmp(argv[1],"--version") == 0){
            char fileName[] = "version";
            outputFile(OUT, fileName);
```

```
            endProgram = true;
        }
        else{
            raiseError("command parameter invalid");
        }
    }

    //MAIN PROGRAM
    while (!endProgram){

        matrix_t* matrix_a;
        matrix_t* matrix_b;
        matrix_t* matrix_c;

        int dimention;
        input = readInput(&dimention);
        matrix_a = create_matrix(dimention,dimention);
        matrix_b = create_matrix(dimention,dimention);
        fillUpMatrices(matrix_a,matrix_b, dimention,input);

        matrix_c = matrix_multiply(matrix_a,matrix_b);
        print_matrix(OUT,matrix_c);

        destroy_matrix(matrix_a);
        destroy_matrix(matrix_b);
        destroy_matrix(matrix_c);

        freeInputArray();
    }
    return 0;
}
```

6. Codigo MIPS32

```
        .file   1 "tp0.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .globl  input
        .globl  input
        .section        .bss
        .align  2
        .type   input, @object
        .size   input, 4
input:
        .space  4
        .text
        .align  2
        .globl  freeInputArray
        .ent    freeInputArray
freeInputArray:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        lw      $v0,input
        beq     $v0,$zero,$L18
        lw      $a0,input
        la      $t9,free
        jal     $ra,$t9
$L18:
        sw      $zero,input
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    freeInputArray
        .size   freeInputArray, .-freeInputArray
        .rdata
        .align  2
$LC0:
        .ascii  "elemento %d: %g\n\000"
        .text
        .align  2
        .globl  printArray
        .ent    printArray
printArray:
        .frame  $fp,48,$ra              # vars= 8, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
```

```
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,48
        .cprestore 16
        sw      $ra,40($sp)
        sw      $fp,36($sp)
        sw      $gp,32($sp)
        move    $fp,$sp
        sw      $a0,48($fp)
        sw      $a1,52($fp)
        sw      $zero,24($fp)
$L20:
        lw      $v0,24($fp)
        lw      $v1,48($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L23
        b       $L19
$L23:
        lw      $v0,24($fp)
        sll     $v1,$v0,3
        lw      $v0,52($fp)
        addu    $v0,$v1,$v0
        la      $a0,$LC0
        lw      $a1,24($fp)
        lw      $a2,0($v0)
        lw      $a3,4($v0)
        la      $t9,printf
        jal     $ra,$t9
        lw      $v0,24($fp)
        addu    $v0,$v0,1
        sw      $v0,24($fp)
        b       $L20
$L19:
        move    $sp,$fp
        lw      $ra,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        j       $ra
        .end    printArray
        .size   printArray, .-printArray
        .rdata
        .align  2
$LC1:
        .ascii  "\n\000"
        .align  2
$LC2:
        .ascii  "==========\n\000"
        .align  2
$LC3:
        .ascii  "ERROR MESSAGE: %s\n\000"
        .text
        .align  2
        .globl  raiseError
        .ent    raiseError
```

```
raiseError:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,40($fp)
        la      $a0,__sF+176
        la      $a1,$LC1
        la      $t9,fprintf
        jal     $ra,$t9
        la      $a0,__sF+176
        la      $a1,$LC2
        la      $t9,fprintf
        jal     $ra,$t9
        la      $a0,__sF+176
        la      $a1,$LC3
        lw      $a2,40($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        la      $a0,__sF+176
        la      $a1,$LC2
        la      $t9,fprintf
        jal     $ra,$t9
        la      $a0,__sF+176
        la      $a1,$LC1
        la      $t9,fprintf
        jal     $ra,$t9
        la      $t9,freeInputArray
        jal     $ra,$t9
        li      $a0,1                   # 0x1
        la      $t9,exit
        jal     $ra,$t9
        .end    raiseError
        .size   raiseError, .-raiseError
        .rdata
        .align  2
$LC4:
        .ascii  "FGETC ERROR: I/O error\000"
        .text
        .align  2
        .globl  readLine
        .ent    readLine
readLine:
        .frame  $fp,64,$ra              # vars= 24, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
```

```
        .set    reorder
        subu    $sp,$sp,64
        .cprestore 16
        sw      $ra,56($sp)
        sw      $fp,52($sp)
        sw      $gp,48($sp)
        move    $fp,$sp
        sw      $a0,64($fp)
        li      $v0,10                  # 0xa
        sw      $v0,24($fp)
        sw      $zero,36($fp)
        move    $a0,$zero
        lw      $a1,24($fp)
        la      $t9,realloc
        jal     $ra,$t9
        sw      $v0,28($fp)
        lw      $v0,28($fp)
        bne     $v0,$zero,$L26
        lw      $v0,28($fp)
        sw      $v0,40($fp)
        b       $L25
$L26:
        .set    noreorder
        nop
        .set    reorder
$L27:
        lw      $a0,64($fp)
        la      $t9,fgetc
        jal     $ra,$t9
        sw      $v0,32($fp)
        lw      $v1,32($fp)
        li      $v0,-1                  # 0xffffffffffffffff
        beq     $v1,$v0,$L28
        lw      $v1,32($fp)
        li      $v0,10                  # 0xa
        bne     $v1,$v0,$L29
        b       $L28
$L29:
        addu    $a1,$fp,36
        lw      $v1,0($a1)
        move    $a0,$v1
        lw      $v0,28($fp)
        addu    $a0,$a0,$v0
        lbu     $v0,32($fp)
        sb      $v0,0($a0)
        addu    $v1,$v1,1
        sw      $v1,0($a1)
        lw      $v1,36($fp)
        lw      $v0,24($fp)
        bne     $v1,$v0,$L27
        lw      $v0,24($fp)
        addu    $v0,$v0,16
        sw      $v0,24($fp)
        lw      $a0,28($fp)
        move    $a1,$v0
        la      $t9,realloc
```

```
        jal     $ra,$t9
        sw      $v0,28($fp)
        lw      $v0,28($fp)
        bne     $v0,$zero,$L27
        lw      $v0,28($fp)
        sw      $v0,40($fp)
        b       $L25
$L28:
        lhu     $v0,__sF+12
        srl     $v0,$v0,6
        andi    $v0,$v0,0x1
        beq     $v0,$zero,$L33
        lw      $a0,28($fp)
        la      $t9,free
        jal     $ra,$t9
        la      $a0,$LC4
        la      $t9,raiseError
        jal     $ra,$t9
$L33:
        addu    $a1,$fp,36
        lw      $v1,0($a1)
        move    $a0,$v1
        lw      $v0,28($fp)
        addu    $v0,$a0,$v0
        sb      $zero,0($v0)
        addu    $v1,$v1,1
        sw      $v1,0($a1)
        lw      $a0,28($fp)
        lw      $a1,36($fp)
        la      $t9,realloc
        jal     $ra,$t9
        sw      $v0,28($fp)
        lw      $v0,28($fp)
        sw      $v0,40($fp)
$L25:
        lw      $v0,40($fp)
        move    $sp,$fp
        lw      $ra,56($sp)
        lw      $fp,52($sp)
        addu    $sp,$sp,64
        j       $ra
        .end    readLine
        .size   readLine, .-readLine
        .rdata
        .align  2
$LC5:
        .ascii  "%g%n\000"
        .align  2
$LC6:
        .ascii  "SSCANF ERROR: I/O error\000"
        .align  2
$LC7:
        .ascii  "No coincide dimension con cantidad de elementos ingresad"
        .ascii  "os\000"
        .align  2
$LC8:
```

```
        .ascii  "Input no numerico\000"
        .text
        .align  2
        .globl  readElementsInLine
        .ent    readElementsInLine
readElementsInLine:
        .frame  $fp,72,$ra              # vars= 32, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,72
        .cprestore 16
        sw      $ra,64($sp)
        sw      $fp,60($sp)
        sw      $gp,56($sp)
        move    $fp,$sp
        sw      $a0,72($fp)
        sw      $a1,76($fp)
        la      $a0,__sF
        la      $t9,readLine
        jal     $ra,$t9
        sw      $v0,24($fp)
        lw      $v0,24($fp)
        sw      $v0,28($fp)
        sw      $zero,40($fp)
$L35:
        addu    $v0,$fp,32
        addu    $v1,$fp,36
        lw      $a0,28($fp)
        la      $a1,$LC5
        move    $a2,$v0
        move    $a3,$v1
        la      $t9,sscanf
        jal     $ra,$t9
        sw      $v0,44($fp)
        lhu     $v0,__sF+12
        srl     $v0,$v0,6
        andi    $v0,$v0,0x1
        beq     $v0,$zero,$L38
        lw      $a0,76($fp)
        la      $t9,free
        jal     $ra,$t9
        lw      $a0,24($fp)
        la      $t9,free
        jal     $ra,$t9
        la      $a0,$LC6
        la      $t9,raiseError
        jal     $ra,$t9
$L38:
        lw      $v1,44($fp)
        li      $v0,1                   # 0x1
        bne     $v1,$v0,$L39
        lw      $v1,28($fp)
        lw      $v0,36($fp)
```

```
        addu    $v0,$v1,$v0
        sw      $v0,28($fp)
        lw      $v0,40($fp)
        sll     $v1,$v0,3
        lw      $v0,76($fp)
        addu    $v0,$v1,$v0
        l.s     $f0,32($fp)
        cvt.d.s $f0,$f0
        s.d     $f0,0($v0)
        lw      $v0,40($fp)
        addu    $v0,$v0,1
        sw      $v0,40($fp)
        b       $L35
$L39:
        lw      $v1,44($fp)
        li      $v0,-1                  # 0xffffffffffffffff
        bne     $v1,$v0,$L40
        lw      $v0,40($fp)
        sw      $v0,48($fp)
        lw      $v1,72($fp)
        lw      $v0,72($fp)
        mult    $v1,$v0
        mflo    $v0
        sll     $v1,$v0,1
        lw      $v0,48($fp)
        beq     $v0,$v1,$L34
        lw      $a0,76($fp)
        la      $t9,free
        jal     $ra,$t9
        lw      $a0,24($fp)
        la      $t9,free
        jal     $ra,$t9
        la      $a0,$LC7
        la      $t9,raiseError
        jal     $ra,$t9
        b       $L34
$L40:
        lw      $v1,44($fp)
        li      $v0,1                   # 0x1
        beq     $v1,$v0,$L35
        lw      $a0,76($fp)
        la      $t9,free
        jal     $ra,$t9
        lw      $a0,24($fp)
        la      $t9,free
        jal     $ra,$t9
        la      $a0,$LC8
        la      $t9,raiseError
        jal     $ra,$t9
$L34:
        move    $sp,$fp
        lw      $ra,64($sp)
        lw      $fp,60($sp)
        addu    $sp,$sp,72
        j       $ra
        .end    readElementsInLine
```

```
        .size   readElementsInLine, .-readElementsInLine
        .rdata
        .align  2
$LC9:
        .ascii  "%g\000"
        .align  2
$LC10:
        .ascii  "FSCANF ERROR: I/O error\000"
        .align  2
$LC11:
        .ascii  "Dimension no numerica\000"
        .align  2
$LC12:
        .ascii  "La dimension no es entera positiva\000"
        .align  2
$LC13:
        .ascii  "No se pudo allocar memoria para inputs\000"
        .text
        .align  2
        .globl  readInput
        .ent    readInput
readInput:
        .frame  $fp,56,$ra              # vars= 16, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,56
        .cprestore 16
        sw      $ra,48($sp)
        sw      $fp,44($sp)
        sw      $gp,40($sp)
        move    $fp,$sp
        sw      $a0,56($fp)
        la      $a0,__sF
        la      $a1,$LC9
        addu    $a2,$fp,24
        la      $t9,fscanf
        jal     $ra,$t9
        sw      $v0,32($fp)
        lw      $v1,32($fp)
        li      $v0,-1                  # 0xffffffffffffffff
        bne     $v1,$v0,$L44
        lhu     $v0,__sF+12
        srl     $v0,$v0,6
        andi    $v0,$v0,0x1
        beq     $v0,$zero,$L45
        la      $a0,$LC10
        la      $t9,raiseError
        jal     $ra,$t9
        b       $L44
$L45:
        move    $a0,$zero
        la      $t9,exit
        jal     $ra,$t9
```

```
$L44:
        lw      $v1,32($fp)
        li      $v0,1                   # 0x1
        beq     $v1,$v0,$L47
        la      $a0,$LC11
        la      $t9,raiseError
        jal     $ra,$t9
$L47:
        l.s     $f0,24($fp)
        trunc.w.s $f0,$f0,$v0
        cvt.s.w $f2,$f0
        l.s     $f0,24($fp)
        sub.s   $f0,$f0,$f2
        s.s     $f0,36($fp)
        l.s     $f2,36($fp)
        mtc1    $zero,$f0
        c.lt.s  $f0,$f2
        bc1t    $L49
        l.s     $f2,24($fp)
        mtc1    $zero,$f0
        c.le.s  $f2,$f0
        bc1t    $L49
        b       $L48
$L49:
        la      $a0,$LC12
        la      $t9,raiseError
        jal     $ra,$t9
$L48:
        lw      $v0,56($fp)
        l.s     $f0,24($fp)
        trunc.w.s $f0,$f0,$v1
        s.s     $f0,0($v0)
        lw      $v0,56($fp)
        lw      $v1,56($fp)
        lw      $a0,0($v0)
        lw      $v0,0($v1)
        mult    $a0,$v0
        mflo    $v0
        sll     $v0,$v0,4
        move    $a0,$v0
        la      $t9,malloc
        jal     $ra,$t9
        sw      $v0,28($fp)
        lw      $v0,28($fp)
        bne     $v0,$zero,$L50
        la      $a0,$LC13
        la      $t9,raiseError
        jal     $ra,$t9
$L50:
        lw      $v0,56($fp)
        lw      $a0,0($v0)
        lw      $a1,28($fp)
        la      $t9,readElementsInLine
        jal     $ra,$t9
        lw      $v0,28($fp)
        move    $sp,$fp
```

```
        lw      $ra,48($sp)
        lw      $fp,44($sp)
        addu    $sp,$sp,56
        j       $ra
        .end    readInput
        .size   readInput, .-readInput
        .rdata
        .align  2
$LC14:
        .ascii  "\000"
        .space  99
        .align  2
$LC15:
        .ascii  "./\000"
        .align  2
$LC16:
        .ascii  "r\000"
        .align  2
$LC17:
        .ascii  "no se pudo abrir archivo de salida\000"
        .align  2
$LC18:
        .ascii  "%c\000"
        .align  2
$LC19:
        .ascii  "FPRINTF ERROR: I/O error\000"
        .text
        .align  2
        .globl  outputFile
        .ent    outputFile
outputFile:
        .frame  $fp,160,$ra             # vars= 120, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,160
        .cprestore 16
        sw      $ra,152($sp)
        sw      $fp,148($sp)
        sw      $gp,144($sp)
        move    $fp,$sp
        sw      $a0,160($fp)
        sw      $a1,164($fp)
        lbu     $v0,$LC14
        sb      $v0,24($fp)
        addu    $v0,$fp,25
        li      $v1,99                  # 0x63
        move    $a0,$v0
        move    $a1,$zero
        move    $a2,$v1
        la      $t9,memset
        jal     $ra,$t9
        addu    $a0,$fp,24
        la      $a1,$LC15
```

```
        la      $t9,strcat
        jal     $ra,$t9
        addu    $a0,$fp,24
        lw      $a1,164($fp)
        la      $t9,strcat
        jal     $ra,$t9
        addu    $a0,$fp,24
        la      $a1,$LC16
        la      $t9,fopen
        jal     $ra,$t9
        sw      $v0,132($fp)
        lw      $v0,132($fp)
        bne     $v0,$zero,$L52
        la      $a0,$LC17
        la      $t9,raiseError
        jal     $ra,$t9
$L52:
        .set    noreorder
        nop
        .set    reorder
$L53:
        lb      $v1,136($fp)
        li      $v0,-1                  # 0xffffffffffffffff
        bne     $v1,$v0,$L55
        b       $L54
$L55:
        lw      $v1,132($fp)
        lw      $v0,132($fp)
        lw      $v0,4($v0)
        addu    $v0,$v0,-1
        sw      $v0,4($v1)
        bgez    $v0,$L56
        lw      $a0,132($fp)
        la      $t9,__srget
        jal     $ra,$t9
        sb      $v0,137($fp)
        b       $L57
$L56:
        lw      $v0,132($fp)
        lw      $v1,0($v0)
        move    $a0,$v1
        lbu     $a0,0($a0)
        sb      $a0,137($fp)
        addu    $v1,$v1,1
        sw      $v1,0($v0)
$L57:
        lbu     $v0,137($fp)
        sb      $v0,136($fp)
        lb      $v0,136($fp)
        lw      $a0,160($fp)
        la      $a1,$LC18
        move    $a2,$v0
        la      $t9,fprintf
        jal     $ra,$t9
        sw      $v0,128($fp)
        lw      $v0,128($fp)
```

```
        bgez    $v0,$L53
        la      $a0,$LC19
        la      $t9,raiseError
        jal     $ra,$t9
        b       $L53
$L54:
        lhu     $v0,__sF+12
        srl     $v0,$v0,6
        andi    $v0,$v0,0x1
        beq     $v0,$zero,$L51
        la      $a0,$LC4
        la      $t9,raiseError
        jal     $ra,$t9
$L51:
        move    $sp,$fp
        lw      $ra,152($sp)
        lw      $fp,148($sp)
        addu    $sp,$sp,160
        j       $ra
        .end    outputFile
        .size   outputFile, .-outputFile
        .rdata
        .align  2
$LC20:
        .ascii  "no se pudo allocar memoria para matriz\000"
        .align  2
$LC21:
        .ascii  "no se pudo allocar memoria para elementos de matriz\000"
        .text
        .align  2
        .globl  create_matrix
        .ent    create_matrix
create_matrix:
        .frame  $fp,48,$ra              # vars= 8, regs= 4/0, args= 16, extra= 8
        .mask   0xd0010000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,48
        .cprestore 16
        sw      $ra,44($sp)
        sw      $fp,40($sp)
        sw      $gp,36($sp)
        sw      $s0,32($sp)
        move    $fp,$sp
        sw      $a0,48($fp)
        sw      $a1,52($fp)
        li      $a0,12                  # 0xc
        la      $t9,malloc
        jal     $ra,$t9
        sw      $v0,24($fp)
        lw      $v0,24($fp)
        bne     $v0,$zero,$L61
        la      $a0,$LC20
        la      $t9,raiseError
```

```
        jal     $ra,$t9
$L61:
        lw      $s0,24($fp)
        lw      $v1,52($fp)
        lw      $v0,48($fp)
        mult    $v1,$v0
        mflo    $v0
        sll     $v0,$v0,3
        move    $a0,$v0
        la      $t9,malloc
        jal     $ra,$t9
        sw      $v0,8($s0)
        lw      $v0,24($fp)
        lw      $v0,8($v0)
        bne     $v0,$zero,$L62
        lw      $a0,24($fp)
        la      $t9,free
        jal     $ra,$t9
        la      $a0,$LC21
        la      $t9,raiseError
        jal     $ra,$t9
$L62:
        lw      $v1,24($fp)
        lw      $v0,48($fp)
        sw      $v0,0($v1)
        lw      $v1,24($fp)
        lw      $v0,52($fp)
        sw      $v0,4($v1)
        lw      $v0,24($fp)
        move    $sp,$fp
        lw      $ra,44($sp)
        lw      $fp,40($sp)
        lw      $s0,32($sp)
        addu    $sp,$sp,48
        j       $ra
        .end    create_matrix
        .size   create_matrix, .-create_matrix
        .align  2
        .globl  destroy_matrix
        .ent    destroy_matrix
destroy_matrix:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,40($fp)
        lw      $v0,40($fp)
        beq     $v0,$zero,$L63
```

```
        lw      $v0,40($fp)
        lw      $a0,8($v0)
        la      $t9,free
        jal     $ra,$t9
        lw      $a0,40($fp)
        la      $t9,free
        jal     $ra,$t9
$L63:
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    destroy_matrix
        .size   destroy_matrix, .-destroy_matrix
        .align  2
        .globl  fillUpMatrices
        .ent    fillUpMatrices
fillUpMatrices:
        .frame  $fp,24,$ra              # vars= 8, regs= 2/0, args= 0, extra= 8
        .mask   0x50000000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,24
        .cprestore 0
        sw      $fp,20($sp)
        sw      $gp,16($sp)
        move    $fp,$sp
        sw      $a0,24($fp)
        sw      $a1,28($fp)
        sw      $a2,32($fp)
        sw      $a3,36($fp)
        sw      $zero,8($fp)
$L66:
        lw      $v1,32($fp)
        lw      $v0,32($fp)
        mult    $v1,$v0
        mflo    $v1
        lw      $v0,8($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L69
        b       $L67
$L69:
        lw      $a0,24($fp)
        lw      $v0,8($fp)
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $a0,$v1,$v0
        lw      $v0,8($fp)
        sll     $v1,$v0,3
        lw      $v0,36($fp)
        addu    $v0,$v1,$v0
        l.d     $f0,0($v0)
        s.d     $f0,0($a0)
```

```
        lw      $v0,8($fp)
        addu    $v0,$v0,1
        sw      $v0,8($fp)
        b       $L66
$L67:
        lw      $v0,32($fp)
        lw      $v1,32($fp)
        mult    $v0,$v1
        mflo    $v0
        sw      $v0,8($fp)
$L70:
        lw      $v1,32($fp)
        lw      $v0,32($fp)
        mult    $v1,$v0
        mflo    $v0
        sll     $v1,$v0,1
        lw      $v0,8($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L73
        b       $L65
$L73:
        lw      $a0,28($fp)
        lw      $v1,32($fp)
        lw      $v0,32($fp)
        mult    $v1,$v0
        mflo    $v1
        lw      $v0,8($fp)
        subu    $v0,$v0,$v1
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $a0,$v1,$v0
        lw      $v0,8($fp)
        sll     $v1,$v0,3
        lw      $v0,36($fp)
        addu    $v0,$v1,$v0
        l.d     $f0,0($v0)
        s.d     $f0,0($a0)
        lw      $v0,8($fp)
        addu    $v0,$v0,1
        sw      $v0,8($fp)
        b       $L70
$L65:
        move    $sp,$fp
        lw      $fp,20($sp)
        addu    $sp,$sp,24
        j       $ra
        .end    fillUpMatrices
        .size   fillUpMatrices, .-fillUpMatrices
        .align  2
        .globl  matrix_multiply
        .ent    matrix_multiply
matrix_multiply:
        .frame  $fp,72,$ra              # vars= 32, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
```

```
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,72
        .cprestore 16
        sw      $ra,64($sp)
        sw      $fp,60($sp)
        sw      $gp,56($sp)
        move    $fp,$sp
        sw      $a0,72($fp)
        sw      $a1,76($fp)
        lw      $v0,72($fp)
        lw      $v0,0($v0)
        sw      $v0,24($fp)
        lw      $a0,24($fp)
        lw      $a1,24($fp)
        la      $t9,create_matrix
        jal     $ra,$t9
        sw      $v0,28($fp)
        sw      $zero,40($fp)
$L75:
        lw      $v1,24($fp)
        lw      $v0,24($fp)
        mult    $v1,$v0
        mflo    $v1
        lw      $v0,40($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L78
        b       $L76
$L78:
        lw      $v1,40($fp)
        lw      $v0,24($fp)
        div     $0,$v1,$v0
        mflo    $v1
        .set    noreorder
        bne     $v0,$0,1f
        nop
        break   7
1:
        .set    reorder
        sw      $v1,32($fp)
        lw      $v1,40($fp)
        lw      $v0,24($fp)
        div     $0,$v1,$v0
        mfhi    $v1
        .set    noreorder
        bne     $v0,$0,1f
        nop
        break   7
1:
        .set    reorder
        sw      $v1,36($fp)
        sw      $zero,48($fp)
        sw      $zero,52($fp)
        sw      $zero,44($fp)
$L79:
        lw      $v0,44($fp)
```

```
        lw      $v1,24($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L82
        b       $L80
$L82:
        lw      $a0,72($fp)
        lw      $v1,32($fp)
        lw      $v0,24($fp)
        mult    $v1,$v0
        mflo    $v1
        lw      $v0,44($fp)
        addu    $v0,$v1,$v0
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $a1,$v1,$v0
        lw      $a0,76($fp)
        lw      $v1,44($fp)
        lw      $v0,24($fp)
        mult    $v1,$v0
        mflo    $v1
        lw      $v0,36($fp)
        addu    $v0,$v1,$v0
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $v0,$v1,$v0
        l.d     $f2,0($a1)
        l.d     $f0,0($v0)
        mul.d   $f2,$f2,$f0
        l.d     $f0,48($fp)
        add.d   $f0,$f0,$f2
        s.d     $f0,48($fp)
        lw      $v0,44($fp)
        addu    $v0,$v0,1
        sw      $v0,44($fp)
        b       $L79
$L80:
        lw      $a0,28($fp)
        lw      $v0,40($fp)
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $v0,$v1,$v0
        l.d     $f0,48($fp)
        s.d     $f0,0($v0)
        lw      $v0,40($fp)
        addu    $v0,$v0,1
        sw      $v0,40($fp)
        b       $L75
$L76:
        lw      $v0,28($fp)
        move    $sp,$fp
        lw      $ra,64($sp)
        lw      $fp,60($sp)
        addu    $sp,$sp,72
        j       $ra
        .end    matrix_multiply
        .size   matrix_multiply, .-matrix_multiply
```

```
        .rdata
        .align  2
$LC22:
        .ascii  "%d\000"
        .text
        .align  2
        .globl  print_matrix
        .ent    print_matrix
print_matrix:
        .frame  $fp,64,$ra              # vars= 24, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,64
        .cprestore 16
        sw      $ra,56($sp)
        sw      $fp,52($sp)
        sw      $gp,48($sp)
        move    $fp,$sp
        sw      $a0,64($fp)
        sw      $a1,68($fp)
        lw      $v0,68($fp)
        lw      $v0,0($v0)
        sw      $v0,24($fp)
        lw      $a0,64($fp)
        la      $a1,$LC22
        lw      $a2,24($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        sw      $v0,44($fp)
        lw      $v0,44($fp)
        bgez    $v0,$L84
        la      $a0,$LC19
        la      $t9,raiseError
        jal     $ra,$t9
$L84:
        lw      $a0,64($fp)
        la      $a1,$LC18
        li      $a2,32                  # 0x20
        la      $t9,fprintf
        jal     $ra,$t9
        sw      $v0,44($fp)
        lw      $v0,44($fp)
        bgez    $v0,$L85
        la      $a0,$LC19
        la      $t9,raiseError
        jal     $ra,$t9
$L85:
        sw      $zero,40($fp)
$L86:
        lw      $v1,24($fp)
        lw      $v0,24($fp)
        mult    $v1,$v0
        mflo    $v1
```

```
        lw      $v0,40($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L89
        b       $L87
$L89:
        lw      $a0,68($fp)
        lw      $v0,40($fp)
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $v0,$v1,$v0
        l.d     $f0,0($v0)
        s.d     $f0,32($fp)
        lw      $a0,64($fp)
        la      $a1,$LC9
        lw      $a2,32($fp)
        lw      $a3,36($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        sw      $v0,44($fp)
        lw      $v0,44($fp)
        bgez    $v0,$L90
        la      $a0,$LC19
        la      $t9,raiseError
        jal     $ra,$t9
$L90:
        lw      $a0,64($fp)
        la      $a1,$LC18
        li      $a2,32                  # 0x20
        la      $t9,fprintf
        jal     $ra,$t9
        sw      $v0,44($fp)
        lw      $v0,44($fp)
        bgez    $v0,$L88
        la      $a0,$LC19
        la      $t9,raiseError
        jal     $ra,$t9
$L88:
        lw      $v0,40($fp)
        addu    $v0,$v0,1
        sw      $v0,40($fp)
        b       $L86
$L87:
        lw      $a0,64($fp)
        la      $a1,$LC1
        la      $t9,fprintf
        jal     $ra,$t9
        sw      $v0,44($fp)
        lw      $v0,44($fp)
        bgez    $v0,$L83
        la      $a0,$LC19
        la      $t9,raiseError
        jal     $ra,$t9
$L83:
        move    $sp,$fp
        lw      $ra,56($sp)
        lw      $fp,52($sp)
```

```
        addu    $sp,$sp,64
        j       $ra
        .end    print_matrix
        .size   print_matrix, .-print_matrix
        .rdata
        .align  2
$LC23:
        .ascii  "-h\000"
        .align  2
$LC24:
        .ascii  "--help\000"
        .align  2
$LC25:
        .ascii  "help\000"
        .align  2
$LC26:
        .ascii  "-V\000"
        .align  2
$LC27:
        .ascii  "--version\000"
        .align  2
$LC28:
        .ascii  "version\000"
        .align  2
$LC29:
        .ascii  "command parameter invalid\000"
        .text
        .align  2
        .globl  main
        .ent    main
main:
        .frame  $fp,80,$ra              # vars= 40, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,80
        .cprestore 16
        sw      $ra,72($sp)
        sw      $fp,68($sp)
        sw      $gp,64($sp)
        move    $fp,$sp
        sw      $a0,80($fp)
        sw      $a1,84($fp)
        la      $v0,__sF+88
        sw      $v0,24($fp)
        sb      $zero,28($fp)
        lw      $v0,80($fp)
        slt     $v0,$v0,2
        bne     $v0,$zero,$L94
        lw      $v0,84($fp)
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC23
        la      $t9,strcmp
```

```
        jal     $ra,$t9
        beq     $v0,$zero,$L96
        lw      $v0,84($fp)
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC24
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L95
$L96:
        lw      $v0,$LC25
        sw      $v0,32($fp)
        lbu     $v0,$LC25+4
        sb      $v0,36($fp)
        addu    $v0,$fp,32
        lw      $a0,24($fp)
        move    $a1,$v0
        la      $t9,outputFile
        jal     $ra,$t9
        li      $v0,1                   # 0x1
        sb      $v0,28($fp)
        b       $L94
$L95:
        lw      $v0,84($fp)
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC26
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L99
        lw      $v0,84($fp)
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC27
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L98
$L99:
        lw      $v0,$LC28
        sw      $v0,40($fp)
        lw      $v0,$LC28+4
        sw      $v0,44($fp)
        addu    $v0,$fp,40
        lw      $a0,24($fp)
        move    $a1,$v0
        la      $t9,outputFile
        jal     $ra,$t9
        li      $v0,1                   # 0x1
        sb      $v0,28($fp)
        b       $L94
$L98:
        la      $a0,$LC29
        la      $t9,raiseError
        jal     $ra,$t9
$L94:
        .set    noreorder
```

```
        nop
        .set    reorder
$L101:
        lbu     $v0,28($fp)
        beq     $v0,$zero,$L103
        b       $L102
$L103:
        addu    $v0,$fp,60
        move    $a0,$v0
        la      $t9,readInput
        jal     $ra,$t9
        sw      $v0,input
        lw      $a0,60($fp)
        lw      $a1,60($fp)
        la      $t9,create_matrix
        jal     $ra,$t9
        sw      $v0,48($fp)
        lw      $a0,60($fp)
        lw      $a1,60($fp)
        la      $t9,create_matrix
        jal     $ra,$t9
        sw      $v0,52($fp)
        lw      $a0,48($fp)
        lw      $a1,52($fp)
        lw      $a2,60($fp)
        lw      $a3,input
        la      $t9,fillUpMatrices
        jal     $ra,$t9
        lw      $a0,48($fp)
        lw      $a1,52($fp)
        la      $t9,matrix_multiply
        jal     $ra,$t9
        sw      $v0,56($fp)
        lw      $a0,24($fp)
        lw      $a1,56($fp)
        la      $t9,print_matrix
        jal     $ra,$t9
        lw      $a0,48($fp)
        la      $t9,destroy_matrix
        jal     $ra,$t9
        lw      $a0,52($fp)
        la      $t9,destroy_matrix
        jal     $ra,$t9
        lw      $a0,56($fp)
        la      $t9,destroy_matrix
        jal     $ra,$t9
        la      $t9,freeInputArray
        jal     $ra,$t9
        b       $L101
$L102:
        move    $v0,$zero
        move    $sp,$fp
        lw      $ra,72($sp)
        lw      $fp,68($sp)
        addu    $sp,$sp,80
        j       $ra
```

```
.end    main
.size   main, .-main
.ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

7. Correcciones

SOBRE PERDIDA DE MEMORIA EN GENERAL
==========

- Se puso como global double* input
- Se agrego una funcion freeInputArray()
- Se agrego que en raiseError() antes de hacer exit() ejecute freeInputArray()

CORECCION 1
==========

- En readLine fuera del while se agrego un if que chequea si ferror() es distinto de 0 y en tal caso llama a raiseError
- En outputFile se hace lo mismo
- Dentro del if que agregue en readLine y antes de llamar a raiseError liberamos el puntero a str a memoria alocada porque se que despues, cuando llame a raiseError, el programa se va a cortar

CORRECCION 2
==========

- En readElementsInLine se agrego head line pointer, porque de lo contrario perdiamos la referencia a line al sumarle el offset.

- Dentro del while de readElementsInLine se cambio los "line"por "head$_{l}ine_{p}ointer"$ $EnreadElementsInLineantesdetirarerro$

CORRECCION 3
==========

- En readElementsInLine vemos si el EOF de fscanf se debia a un error o no con la funcion ferror.

CORRECCION 4
==========

- En la funcion print matrix y en la funcion outputFile se agrego la variable return$_{v}alue$. $Enlafuncionprintmatrixyenlafuncionoutput Filesecambiantodoslosfprintfporunavalidacionqueusaferror.$

VARIOS
==========

- Se agrego para que cuando uno ingresa un parametro de comando invalido tire error
- Se agrega if (ferror(stdin) != 0)raiseError("SSCANF ERROR: I/O error"); abajo del sscanf en readElementsInLine()

8. Enunciado

Se adjunta el enunciado del trábajo práctico 0. [pages=-,pagecommand=,width=]tp0-2019-2q.pdf