Trabajo Práctico 0: Infraestructura básica

Duhalde, Agustín, *Padrón Nro. 97.063*
rbnm.jimenez@gmail.com

Barrabasqui, Horacio, *Padrón Nro. 78.407*
felixcarp@gmail.com

Suárez, Emiliano, *Padrón Nro. 78.372*
emilianosuarez@gmail.com

Fecha Entrega: *05/04/2016*

1er. Cuatrimestre de 2016
66.20 Organización de Computadoras – Práctica Martes
Facultad de Ingeniería, Universidad de Buenos Aires

Resumen

Se implementó un programa en lenguaje **C**, que multiplica matrices cuadradas de números reales, representados en punto flotante de doble precisión.

1. Introducción

En este Trabajo Práctico se pretende familiarizarse con las herramientas de software que usaremos en los siguientes trabajos.

Además, se utilizará *GXemul* para simular una máquina *MIPS* corriendo una versión reciente del sistema operativo *NetBSD*.

El programa implementado, muestra por *stdout* u *output file*, el resultado de multiplicar dos matrices cuadras (puede ser mas de una operación), previamente leídas por *stdin* o *input file*.

2. Diseño e Implementación

Se debió respetar la interfaz mostrada en el enunciado, tanto para la estructura de las matrices, como para la implementación de ciertos métodos.

El comando acepta 2 parámetros para mostrar la Ayuda y la Versión del programa:

```
$ ./tp0 -h
$ ./tp0 --help
```

Para desplegar la ayuda del comando. Y los siguientes comandos para mostrar la versión:

```
$ ./tp0 -V
$ ./tp0 --version
```

Inicialmente el programa revisa la cadena de parametros ingresada y determina que tipo operación debe realizar.

En caso de no recibir los parámetros antes mencionados, se lee desde los argumentos, para saber cuales son los archivos de entrada (opcional) y el de salida.

Las matrices a multiplicar, como así también la resultado del producto, se representan en el formato row major order (para más información, puede consultarse la bibliografía mencionada al final del informe).

3. Comando para compilar el programa

Para compilar el programa se debe abrir una terminal en la carpeta donde están alojados los archivos fuentes (*src/*) y se ejecuta el siguiente comando:

```
../src$ make
```

Para generar el ejecutable *tp0*.

make: se encargara de compilar los archivos generando el ejecutable.

Se creó un archivo *Makefile* que permite compilar tanto la versión en *MIPS*, como en la versión implementada completamente en *C*.

El *Makefile* puede observarse a continuación:

```
CC=gcc -g -O0
CFLAGS=-c -Wall
LDFLAGS=
OBJ_DIR=/
SOURCES=main.c

OBJECTS=$(SOURCES:.c=.o)
EXECUTABLE=tp0

all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
	$(CC) $(LDFLAGS) $(OBJECTS) -o $@

.c.o:
	$(CC) $(CFLAGS) $< -o $@

clean:
	rm -rf $(EXECUTABLE) $(OBJECTS)
```

../src/Makefile

4. Casos de Prueba

Algunos de los casos de pruebas realizados, pueden observarse a continuación:
Importante: Los resultados fueron modificados a 2 decimales por prolijidad del informe. En la corrida real, los mismos se verán con 6 decimales.

```
root@:~/2016/tp0# cat varias.txt
3 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
3 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5 6 6 6
2 1 2 3 4 5 6 7 8
root@:~/2016/tp0# cat output.txt
root@:~/2016/tp0# ./tp0 varias.txt output.txt
root@:~/2016/tp0# cat output.txt
3 84.00 90.00 96.00 201.00 216.00 231.00 318.00 342.00 366.00
3 15.00 15.00 15.00 30.00 30.00 30.00 45.00 45.00 45.00
2 19.00 22.00 43.00 50.00

root@:~/2016/tp0# cat example.txt
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
root@:~/2016/tp0# cat example.txt | ./tp0
2 7.00 10.00 15.00 22.00
3 1.00 2.00 3.00 4.00 5.00 6.100 3.00 2.00 1.00

root@:~/2016/tp0# ./tp0
2 1 2 3 4 1 2 3 4
2 7.00 10.00 15.00 22.00
```

```
3 1 2 3 4 5 6.1 3 2 1 1 0 0 0 1 0 0 0 1
3 1.00 2.00 3.00 4.00 5.00 6.10000 3.00 2.00 1.00

root@:~/2016/tp0# cat output.txt
root@:~/2016/tp0# cat example.txt
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
root@:~/2016/tp0# cat example.txt | ./tp0 output.txt
root@:~/2016/tp0# cat output.txt
2 7.00 10.00 15.00 22.00
3 1.00 2.00 3.00 4.00 5.00 6.100 3.00 2.00 1.00
```

5. Conclusiones

El presente trabajo permitió la familiarización con las herramientas de compilación de código C y código assembly en un entorno que emula la arquitectura MIPS 32, asegurando la portabilidad del programa. Es decir, el programa pudo se ejecutado tanto en un entorno UNIX, como en el NetBSD con los mismos resultados.

6. Apéndice

6.1. Código Fuente: main.c

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

#define ERROR_OPEN_FILE 10

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

// Constructor de matrix_t
matrix_t* create_matrix(size_t rows, size_t cols);

// Destructor de matrix_t
void destroy_matrix(matrix_t* m);

// Imprime matrix_t sobre el file pointer fp en el formato solicitado
// por el enunciado
int print_matrix(FILE* fp, matrix_t* m);

// Multiplica las matrices en m1 y m2
matrix_t* matrix_multiply(matrix_t* m1, matrix_t* m2);

// Lee una linea por stdin y devuelve los elementos de la matriz y la
// dimension por referencia.
double* read_arguments(char* line, int line_size,
                       int *matrix_dimension);

// Devuele la cantidad total de elementos de ambas matrices.
int get_amount_element(int dimension);

// Devuelve la cantidad de elementos de una matriz.
int get_matrix_elements(int dimension);

void printHelp();
void printVersion();
int readFromStdInput(int argumentCount);

int matrices_multiply(FILE* input, FILE* output);

int main (int argc, char *argv[])
{
    char* param;
    int result = 0;
    FILE* input;
    FILE* output;

    // Valores por defecto
    input = stdin;
    output = stdout;

    if (readFromStdInput(argc)) {
        // Leo desde stdin
        result = matrices_multiply(input, output);
```

```
57
58      } else if (argc >= 2) { // Parseo los argumentos
59          param = *(argv + 1);
60          if ((strcmp(param, "-h") == 0)
61              || (strcmp(param, "--help") == 0) ) {
62              printHelp();
63              return 0;
64          }
65          else if ((strcmp(param, "-V") == 0)
66                  || (strcmp(param, "--version") == 0)) {
67              printVersion();
68              return 0;
69          }
70          else if (argc == 2) {
71              // Tengo archivo de output unicamente.
72              if (NULL == (output = fopen(argv[1], "w+"))) {
73                  fprintf(stderr,
74                          "Output File '%s' doesn't exist.\n", argv
                                [1]);
75                  exit(ERROR_OPEN_FILE);
76              }
77          }
78          else {
79              // Tengo archivo de input y output
80              if (NULL == (input = fopen(argv[1], "r"))) {
81                  fprintf(stderr,
82                          "Input File '%s' doesn't exist.\n", argv
                                [1]);
83                  exit(ERROR_OPEN_FILE);
84              }
85              if (NULL == (output = fopen(argv[2], "w+"))) {
86                  fprintf(stderr,
87                          "Output File '%s' doesn't exist.\n", argv
                                [2]);
88                  exit(ERROR_OPEN_FILE);
89              }
90          }
91          result = matrices_multiply(input, output);
92      }
93
94      return result;
95  }
96
97  // Constructor de matrix_t
98  matrix_t* create_matrix(size_t rows, size_t cols)
99  {
100     matrix_t* new_matrix = malloc(sizeof(matrix_t));
101     new_matrix->rows = rows;
102     new_matrix->cols = cols;
103
104     return new_matrix;
105 }
106
107 // Destructor de matrix_t
108 void destroy_matrix(matrix_t* m)
109 {
110     free(m);
111 }
112
113 // Imprime matrix_t sobre el file pointer fp en el formato
        solicitado
114 // por el enunciado
```

```
115 int print_matrix(FILE* fp, matrix_t* m)
116 {
117     int i = 0;
118     int dim = m->rows;
119 
120     fprintf(fp, "%d", dim);
121     for (i = 0; i < dim * dim; i++) {
122         fprintf(fp, " %f", m->array[i]);
123     }
124     fprintf(fp, "\n");
125     return 0;
126 }
127 
128 matrix_t* matrix_multiply(matrix_t* m1, matrix_t* m2)
129 {
130     matrix_t* result = create_matrix(m1->rows, m1->cols);
131     result->array = malloc(m1->rows * m1->cols * sizeof(double));
132     double temp;
133     int dim = m1->rows;
134     int i = 0;
135     int j = 0;
136     int k = 0;
137     int x = 0;
138 
139     for(; k < (dim * dim) ; i += dim){
140         for(; x < dim; j++, k++, x++){
141             temp = 0;
142 
143             for (;j < (dim * dim); i++, j += dim){
144                 temp = temp + m1->array[i] * m2->array[j];
145             }
146 
147             j -= (dim * dim);
148             i -= dim;
149             result->array[k] = temp;
150         }
151         x = 0;
152         j = 0;
153     }
154 
155     return result;
156 }
157 
158 double* read_arguments(char* line, int line_size,
159                        int *matrix_dimension) {
160     char *first_token;
161     char *search = " ";
162     double element = 0;
163     double* matrix_elements;
164     double* pointer_to_first_element;
165     int number_of_elements = 0;
166 
167      // Leo la dimension de la matriz, que esta en la primer
           posicion
168      // de la linea.
169     first_token = strtok(line, search);
170     *matrix_dimension = atoi(first_token);
171     number_of_elements = (get_amount_element(*matrix_dimension));
172     matrix_elements = malloc(number_of_elements * sizeof(double));
173     pointer_to_first_element = matrix_elements;
174 
175     while ( (first_token = strtok( NULL, search)) != NULL) {
```

```
176         element = atof(first_token);
177         *matrix_elements = element;
178         matrix_elements++;
179     }
180 
181     // Seteo el fin del array
182     *matrix_elements = '\0';
183 
184     matrix_elements = pointer_to_first_element;
185 
186     return matrix_elements;
187 }
188 
189 int get_amount_element(int dimension)
190 {
191     return dimension * dimension * 2;
192 }
193 
194 int get_matrix_elements(int dimension)
195 {
196     return dimension * dimension;
197 }
198 int readFromStdInput(int argumentCount) {
199     return (int)(argumentCount < 2);
200 }
201 
202 void printHelp()
203 {
204     fprintf(stdout, "$ tp0 -h\n");
205     fprintf(stdout, "Usage:\n");
206     fprintf(stdout, "  tp0 -h\n");
207     fprintf(stdout, "  tp0 -V\n");
208     fprintf(stdout, "  tp0 < in_file > out_file\n");
209     fprintf(stdout, "Options:\n");
210     fprintf(stdout, "  -V, --version    Print version and quit.\n")
        ;
211     fprintf(stdout, "  -h, --help       ");
212     fprintf(stdout, "Print this information and quit.\n\n");
213     fprintf(stdout, "Examples:\n");
214     fprintf(stdout, "  tp0 < in.txt > out.txt\n");
215     fprintf(stdout, "  cat in.txt | tp0 > out.txt\n\n");
216 }
217 
218 void printVersion()
219 {
220     fprintf(stdout, "Copyright (c) 2016\n");
221     fprintf(stdout, "MIPS - Infraestructura basica. v1.0.0\n\n");
222 }
223 
224 int matrices_multiply(FILE* input, FILE* output)
225 {
226     char c;
227     char* line = malloc(sizeof(char));
228     int chars_per_line = 0;
229     int matrix_dimension = 0;
230     int amount_elements = 0;
231     int elements_per_matrix = 0;
232     int print_result = 0;
233     double* element_pointer;
234     double* m1_elements;
235     double* m2_elements;
236     matrix_t* m1;
```

```
237     matrix_t* m2;
238     matrix_t* product_matrix;
239 
240     while( (c = getc(input)) != EOF)
241     {
242         chars_per_line++;
243 
244         // Incremento la cantidad de memoria en funcion de la
                cantidad
245         // de caracteres de la linea.
246         // Luego, asigno el caracter a la linea
247         line = realloc(line, chars_per_line * sizeof(char));
248         line[chars_per_line - 1] = c;
249 
250         // Si finalizo la linea, multiplico las matrices
251         if ('\n' == c) {
252             element_pointer = read_arguments(line, chars_per_line,
253                                              &matrix_dimension);
254 
255             // Calculo la cantidad de elementos de ambas matrices.
256             amount_elements = get_amount_element(matrix_dimension);
257             elements_per_matrix = get_matrix_elements(
                    matrix_dimension);
258 
259             // Seteo los arrays de elementos para ambas matrices
260             m1_elements = element_pointer;
261             m2_elements = &element_pointer[elements_per_matrix];
262 
263             m1 = create_matrix(matrix_dimension, matrix_dimension);
264             m1->array = m1_elements;
265 
266             m2 = create_matrix(matrix_dimension, matrix_dimension);
267             m2->array = m2_elements;
268 
269             product_matrix = matrix_multiply(m1, m2);
270             print_result = print_matrix(output, product_matrix);
271 
272             destroy_matrix(m1);
273             destroy_matrix(m2);
274             destroy_matrix(product_matrix);
275             free(element_pointer);
276 
277             chars_per_line = 0;
278         }
279     }
280 
281     free(line);
282 
283     return print_result;
284 }
```

../src/main.c

6.2. main.s

```
1   .file 1 "main.c"
2   .section .mdebug.abi32
3   .previous
4   .abicalls
5   .rdata
6   .align 2
7 $LC0:
8   .ascii "-h\000"
9   .align 2
10 $LC1:
11  .ascii "--help\000"
12  .align 2
13 $LC2:
14  .ascii "-V\000"
15  .align 2
16 $LC3:
17  .ascii "--version\000"
18  .align 2
19 $LC4:
20  .ascii "w+\000"
21  .align 2
22 $LC5:
23  .ascii "Output File '%s' doesn't exist.\n\000"
24  .align 2
25 $LC6:
26  .ascii "r\000"
27  .align 2
28 $LC7:
29  .ascii "Input File '%s' doesn't exist.\n\000"
30  .text
31  .align 2
32  .globl main
33  .ent main
34 main:
35  .frame $fp,64,$31    # vars= 24, regs= 3/0, args= 16, extra= 8
36  .mask 0xd0000000,-8
37  .fmask 0x00000000,0
38  .set noreorder
39  .cpload $25
40  .set reorder
41  subu $sp,$sp,64
42  .cprestore 16
43  sw $31,56($sp)
44  sw $fp,52($sp)
45  sw $28,48($sp)
46  move $fp,$sp
47  sw $4,64($fp)
48  sw $5,68($fp)
49  sw $0,28($fp)
50  la $2,__sF
51  sw $2,32($fp)
52  la $2,__sF+88
53  sw $2,36($fp)
54  lw $4,64($fp)
55  la $25,readFromStdInput
56  jal $31,$25
57  beq $2,$0,$L18
58  lw $4,32($fp)
59  lw $5,36($fp)
60  la $25,matrices_multiply
```

```
61    jal $31,$25
62    sw  $2,28($fp)
63    b $L19
64  $L18:
65    lw  $2,64($fp)
66    slt $2,$2,2
67    bne $2,$0,$L19
68    lw  $2,68($fp)
69    addu  $2,$2,4
70    lw  $2,0($2)
71    sw  $2,24($fp)
72    lw  $4,24($fp)
73    la  $5,$LC0
74    la  $25,strcmp
75    jal $31,$25
76    beq $2,$0,$L22
77    lw  $4,24($fp)
78    la  $5,$LC1
79    la  $25,strcmp
80    jal $31,$25
81    bne $2,$0,$L21
82  $L22:
83    la  $25,printHelp
84    jal $31,$25
85    sw  $0,40($fp)
86    b $L17
87  $L21:
88    lw  $4,24($fp)
89    la  $5,$LC2
90    la  $25,strcmp
91    jal $31,$25
92    beq $2,$0,$L25
93    lw  $4,24($fp)
94    la  $5,$LC3
95    la  $25,strcmp
96    jal $31,$25
97    bne $2,$0,$L24
98  $L25:
99    la  $25,printVersion
100   jal $31,$25
101   sw  $0,40($fp)
102   b $L17
103 $L24:
104   lw  $3,64($fp)
105   li  $2,2      # 0x2
106   bne $3,$2,$L27
107   lw  $2,68($fp)
108   addu  $2,$2,4
109   lw  $4,0($2)
110   la  $5,$LC4
111   la  $25,fopen
112   jal $31,$25
113   sw  $2,36($fp)
114   lw  $2,36($fp)
115   bne $2,$0,$L23
116   lw  $2,68($fp)
117   addu  $2,$2,4
118   la  $4,__sF+176
119   la  $5,$LC5
120   lw  $6,0($2)
121   la  $25,fprintf
122   jal $31,$25
```

```
123   li   $4,10      # 0xa
124   la   $25,exit
125   jal  $31,$25
126 $L27:
127   lw   $2,68($fp)
128   addu   $2,$2,4
129   lw   $4,0($2)
130   la   $5,$LC6
131   la   $25,fopen
132   jal  $31,$25
133   sw   $2,32($fp)
134   lw   $2,32($fp)
135   bne  $2,$0,$L30
136   lw   $2,68($fp)
137   addu   $2,$2,4
138   la   $4,__sF+176
139   la   $5,$LC7
140   lw   $6,0($2)
141   la   $25,fprintf
142   jal  $31,$25
143   li   $4,10      # 0xa
144   la   $25,exit
145   jal  $31,$25
146 $L30:
147   lw   $2,68($fp)
148   addu   $2,$2,8
149   lw   $4,0($2)
150   la   $5,$LC4
151   la   $25,fopen
152   jal  $31,$25
153   sw   $2,36($fp)
154   lw   $2,36($fp)
155   bne  $2,$0,$L23
156   lw   $2,68($fp)
157   addu   $2,$2,8
158   la   $4,__sF+176
159   la   $5,$LC5
160   lw   $6,0($2)
161   la   $25,fprintf
162   jal  $31,$25
163   li   $4,10      # 0xa
164   la   $25,exit
165   jal  $31,$25
166 $L23:
167   lw   $4,32($fp)
168   lw   $5,36($fp)
169   la   $25,matrices_multiply
170   jal  $31,$25
171   sw   $2,28($fp)
172 $L19:
173   lw   $2,28($fp)
174   sw   $2,40($fp)
175 $L17:
176   lw   $2,40($fp)
177   move   $sp,$fp
178   lw   $31,56($sp)
179   lw   $fp,52($sp)
180   addu   $sp,$sp,64
181   j $31
182   .end   main
183   .size main, .-main
184   .align   2
```

```
185     .globl  create_matrix
186     .ent    create_matrix
187 create_matrix:
188     .frame  $fp,48,$31      # vars= 8, regs= 3/0, args= 16, extra= 8
189     .mask   0xd0000000,-8
190     .fmask  0x00000000,0
191     .set    noreorder
192     .cpload $25
193     .set    reorder
194     subu    $sp,$sp,48
195     .cprestore 16
196     sw      $31,40($sp)
197     sw      $fp,36($sp)
198     sw      $28,32($sp)
199     move    $fp,$sp
200     sw      $4,48($fp)
201     sw      $5,52($fp)
202     li      $4,12       # 0xc
203     la      $25,malloc
204     jal     $31,$25
205     sw      $2,24($fp)
206     lw      $3,24($fp)
207     lw      $2,48($fp)
208     sw      $2,0($3)
209     lw      $3,24($fp)
210     lw      $2,52($fp)
211     sw      $2,4($3)
212     lw      $2,24($fp)
213     move    $sp,$fp
214     lw      $31,40($sp)
215     lw      $fp,36($sp)
216     addu    $sp,$sp,48
217     j       $31
218     .end    create_matrix
219     .size   create_matrix, .-create_matrix
220     .align  2
221     .globl  destroy_matrix
222     .ent    destroy_matrix
223 destroy_matrix:
224     .frame  $fp,40,$31      # vars= 0, regs= 3/0, args= 16, extra= 8
225     .mask   0xd0000000,-8
226     .fmask  0x00000000,0
227     .set    noreorder
228     .cpload $25
229     .set    reorder
230     subu    $sp,$sp,40
231     .cprestore 16
232     sw      $31,32($sp)
233     sw      $fp,28($sp)
234     sw      $28,24($sp)
235     move    $fp,$sp
236     sw      $4,40($fp)
237     lw      $4,40($fp)
238     la      $25,free
239     jal     $31,$25
240     move    $sp,$fp
241     lw      $31,32($sp)
242     lw      $fp,28($sp)
243     addu    $sp,$sp,40
244     j       $31
245     .end    destroy_matrix
246     .size   destroy_matrix, .-destroy_matrix
```

```
247     .rdata
248     .align  2
249 $LC8:
250     .ascii  "%d\000"
251     .align  2
252 $LC9:
253     .ascii  " %f\000"
254     .align  2
255 $LC10:
256     .ascii  "\n\000"
257     .text
258     .align  2
259     .globl  print_matrix
260     .ent    print_matrix
261 print_matrix:
262     .frame  $fp,48,$31      # vars= 8, regs= 3/0, args= 16, extra= 8
263     .mask   0xd0000000,-8
264     .fmask  0x00000000,0
265     .set    noreorder
266     .cpload $25
267     .set    reorder
268     subu    $sp,$sp,48
269     .cprestore 16
270     sw      $31,40($sp)
271     sw      $fp,36($sp)
272     sw      $28,32($sp)
273     move    $fp,$sp
274     sw      $4,48($fp)
275     sw      $5,52($fp)
276     sw      $0,24($fp)
277     lw      $2,52($fp)
278     lw      $2,0($2)
279     sw      $2,28($fp)
280     lw      $4,48($fp)
281     la      $5,$LC8
282     lw      $6,28($fp)
283     la      $25,fprintf
284     jal     $31,$25
285     sw      $0,24($fp)
286 $L35:
287     lw      $3,28($fp)
288     lw      $2,28($fp)
289     mult    $3,$2
290     mflo    $3
291     lw      $2,24($fp)
292     slt     $2,$2,$3
293     bne     $2,$0,$L38
294     b       $L36
295 $L38:
296     lw      $4,52($fp)
297     lw      $2,24($fp)
298     sll     $3,$2,3
299     lw      $2,8($4)
300     addu    $2,$3,$2
301     lw      $4,48($fp)
302     la      $5,$LC9
303     lw      $6,0($2)
304     lw      $7,4($2)
305     la      $25,fprintf
306     jal     $31,$25
307     lw      $2,24($fp)
308     addu    $2,$2,1
```

```
309   sw  $2,24($fp)
310   b $L35
311 $L36:
312   lw  $4,48($fp)
313   la  $5,$LC10
314   la  $25,fprintf
315   jal $31,$25
316   move  $2,$0
317   move  $sp,$fp
318   lw  $31,40($sp)
319   lw  $fp,36($sp)
320   addu  $sp,$sp,48
321   j $31
322   .end  print_matrix
323   .size print_matrix, .-print_matrix
324   .align  2
325   .globl  matrix_multiply
326   .ent  matrix_multiply
327 matrix_multiply:
328   .frame  $fp,80,$31    # vars= 40, regs= 4/0, args= 16, extra= 8
329   .mask 0xd0010000,-4
330   .fmask  0x00000000,0
331   .set  noreorder
332   .cpload $25
333   .set  reorder
334   subu  $sp,$sp,80
335   .cprestore 16
336   sw  $31,76($sp)
337   sw  $fp,72($sp)
338   sw  $28,68($sp)
339   sw  $16,64($sp)
340   move  $fp,$sp
341   sw  $4,80($fp)
342   sw  $5,84($fp)
343   lw  $2,80($fp)
344   lw  $3,80($fp)
345   lw  $4,0($2)
346   lw  $5,4($3)
347   la  $25,create_matrix
348   jal $31,$25
349   sw  $2,24($fp)
350   lw  $16,24($fp)
351   lw  $2,80($fp)
352   lw  $3,80($fp)
353   lw  $4,0($2)
354   lw  $2,4($3)
355   mult  $4,$2
356   mflo  $2
357   sll $2,$2,3
358   move  $4,$2
359   la  $25,malloc
360   jal $31,$25
361   sw  $2,8($16)
362   lw  $2,80($fp)
363   lw  $2,0($2)
364   sw  $2,40($fp)
365   sw  $0,44($fp)
366   sw  $0,48($fp)
367   sw  $0,52($fp)
368   sw  $0,56($fp)
369 $L40:
370   lw  $3,40($fp)
```

```
371     lw   $2,40($fp)
372     mult   $3,$2
373     mflo   $3
374     lw   $2,52($fp)
375     slt  $2,$2,$3
376     bne  $2,$0,$L44
377     b  $L41
378 $L44:
379     lw   $2,56($fp)
380     lw   $3,40($fp)
381     slt  $2,$2,$3
382     bne  $2,$0,$L47
383     b  $L45
384 $L47:
385     sw   $0,32($fp)
386     sw   $0,36($fp)
387 $L48:
388     lw   $3,40($fp)
389     lw   $2,40($fp)
390     mult   $3,$2
391     mflo   $3
392     lw   $2,48($fp)
393     slt  $2,$2,$3
394     bne  $2,$0,$L51
395     b  $L49
396 $L51:
397     lw   $4,80($fp)
398     lw   $2,44($fp)
399     sll  $3,$2,3
400     lw   $2,8($4)
401     addu   $5,$3,$2
402     lw   $4,84($fp)
403     lw   $2,48($fp)
404     sll  $3,$2,3
405     lw   $2,8($4)
406     addu   $2,$3,$2
407     l.d  $f2,0($5)
408     l.d  $f0,0($2)
409     mul.d  $f2,$f2,$f0
410     l.d  $f0,32($fp)
411     add.d  $f0,$f0,$f2
412     s.d  $f0,32($fp)
413     lw   $2,44($fp)
414     addu   $2,$2,1
415     sw   $2,44($fp)
416     lw   $3,48($fp)
417     lw   $2,40($fp)
418     addu   $2,$3,$2
419     sw   $2,48($fp)
420     b  $L48
421 $L49:
422     lw   $3,40($fp)
423     lw   $2,40($fp)
424     mult   $3,$2
425     mflo   $3
426     lw   $2,48($fp)
427     subu   $2,$2,$3
428     sw   $2,48($fp)
429     lw   $3,44($fp)
430     lw   $2,40($fp)
431     subu   $2,$3,$2
432     sw   $2,44($fp)
```

```
  lw  $4,24($fp)
  lw  $2,52($fp)
  sll $3,$2,3
  lw  $2,8($4)
  addu  $2,$3,$2
  l.d $f0,32($fp)
  s.d $f0,0($2)
  lw  $2,48($fp)
  addu  $2,$2,1
  sw  $2,48($fp)
  lw  $2,52($fp)
  addu  $2,$2,1
  sw  $2,52($fp)
  lw  $2,56($fp)
  addu  $2,$2,1
  sw  $2,56($fp)
  b $L44
$L45:
  sw  $0,56($fp)
  sw  $0,48($fp)
  lw  $2,44($fp)
  lw  $3,40($fp)
  addu  $2,$2,$3
  sw  $2,44($fp)
  b $L40
$L41:
  lw  $2,24($fp)
  move  $sp,$fp
  lw  $31,76($sp)
  lw  $fp,72($sp)
  lw  $16,64($sp)
  addu  $sp,$sp,80
  j $31
  .end  matrix_multiply
  .size matrix_multiply, .-matrix_multiply
  .rdata
  .align  2
$LC11:
  .ascii  " \000"
  .text
  .align  2
  .globl  read_arguments
  .ent  read_arguments
read_arguments:
  .frame  $fp,72,$31    # vars= 32, regs= 3/0, args= 16, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,72
  .cprestore 16
  sw  $31,64($sp)
  sw  $fp,60($sp)
  sw  $28,56($sp)
  move  $fp,$sp
  sw  $4,72($fp)
  sw  $5,76($fp)
  sw  $6,80($fp)
  la  $2,$LC11
  sw  $2,28($fp)
  sw  $0,32($fp)
```

```
495   sw   $0,36($fp)
496   sw   $0,48($fp)
497   lw   $4,72($fp)
498   lw   $5,28($fp)
499   la   $25,strtok
500   jal  $31,$25
501   sw   $2,24($fp)
502   lw   $4,24($fp)
503   la   $25,atoi
504   jal  $31,$25
505   move   $3,$2
506   lw   $2,80($fp)
507   sw   $3,0($2)
508   lw   $2,80($fp)
509   lw   $4,0($2)
510   la   $25,get_amount_element
511   jal  $31,$25
512   sw   $2,48($fp)
513   lw   $2,48($fp)
514   sll  $2,$2,3
515   move   $4,$2
516   la   $25,malloc
517   jal  $31,$25
518   sw   $2,40($fp)
519   lw   $2,40($fp)
520   sw   $2,44($fp)
521 $L53:
522   move   $4,$0
523   lw   $5,28($fp)
524   la   $25,strtok
525   jal  $31,$25
526   sw   $2,24($fp)
527   lw   $2,24($fp)
528   bne  $2,$0,$L55
529   b  $L54
530 $L55:
531   lw   $4,24($fp)
532   la   $25,atof
533   jal  $31,$25
534   s.d  $f0,32($fp)
535   lw   $2,40($fp)
536   l.d  $f0,32($fp)
537   s.d  $f0,0($2)
538   lw   $2,40($fp)
539   addu   $2,$2,8
540   sw   $2,40($fp)
541   b  $L53
542 $L54:
543   lw   $2,40($fp)
544   sw   $0,0($2)
545   sw   $0,4($2)
546   lw   $2,44($fp)
547   sw   $2,40($fp)
548   lw   $2,40($fp)
549   move   $sp,$fp
550   lw   $31,64($sp)
551   lw   $fp,60($sp)
552   addu   $sp,$sp,72
553   j  $31
554   .end   read_arguments
555   .size read_arguments, .-read_arguments
556   .align   2
```

```
557     .globl  get_amount_element
558     .ent    get_amount_element
559 get_amount_element:
560     .frame  $fp,16,$31      # vars= 0, regs= 2/0, args= 0, extra= 8
561     .mask   0x50000000,-4
562     .fmask  0x00000000,0
563     .set    noreorder
564     .cpload $25
565     .set    reorder
566     subu    $sp,$sp,16
567     .cprestore 0
568     sw      $fp,12($sp)
569     sw      $28,8($sp)
570     move    $fp,$sp
571     sw      $4,16($fp)
572     lw      $3,16($fp)
573     lw      $2,16($fp)
574     mult    $3,$2
575     mflo    $2
576     sll     $2,$2,1
577     move    $sp,$fp
578     lw      $fp,12($sp)
579     addu    $sp,$sp,16
580     j       $31
581     .end    get_amount_element
582     .size   get_amount_element, .-get_amount_element
583     .align  2
584     .globl  get_matrix_elements
585     .ent    get_matrix_elements
586 get_matrix_elements:
587     .frame  $fp,16,$31      # vars= 0, regs= 2/0, args= 0, extra= 8
588     .mask   0x50000000,-4
589     .fmask  0x00000000,0
590     .set    noreorder
591     .cpload $25
592     .set    reorder
593     subu    $sp,$sp,16
594     .cprestore 0
595     sw      $fp,12($sp)
596     sw      $28,8($sp)
597     move    $fp,$sp
598     sw      $4,16($fp)
599     lw      $3,16($fp)
600     lw      $2,16($fp)
601     mult    $3,$2
602     mflo    $2
603     move    $sp,$fp
604     lw      $fp,12($sp)
605     addu    $sp,$sp,16
606     j       $31
607     .end    get_matrix_elements
608     .size   get_matrix_elements, .-get_matrix_elements
609     .align  2
610     .globl  readFromStdInput
611     .ent    readFromStdInput
612 readFromStdInput:
613     .frame  $fp,16,$31      # vars= 0, regs= 2/0, args= 0, extra= 8
614     .mask   0x50000000,-4
615     .fmask  0x00000000,0
616     .set    noreorder
617     .cpload $25
618     .set    reorder
```

```
619   subu  $sp,$sp,16
620   .cprestore 0
621   sw  $fp,12($sp)
622   sw  $28,8($sp)
623   move  $fp,$sp
624   sw  $4,16($fp)
625   lw  $2,16($fp)
626   slt $2,$2,2
627   move  $sp,$fp
628   lw  $fp,12($sp)
629   addu  $sp,$sp,16
630   j $31
631   .end  readFromStdInput
632   .size readFromStdInput, .-readFromStdInput
633   .rdata
634   .align  2
635 $LC12:
636   .ascii  "$ tp0 -h\n\000"
637   .align  2
638 $LC13:
639   .ascii  "Usage:\n\000"
640   .align  2
641 $LC14:
642   .ascii  "  tp0 -h\n\000"
643   .align  2
644 $LC15:
645   .ascii  "  tp0 -V\n\000"
646   .align  2
647 $LC16:
648   .ascii  "  tp0 < in_file > out_file\n\000"
649   .align  2
650 $LC17:
651   .ascii  "Options:\n\000"
652   .align  2
653 $LC18:
654   .ascii  "  -V, --version    Print version and quit.\n\000"
655   .align  2
656 $LC19:
657   .ascii  "  -h, --help       \000"
658   .align  2
659 $LC20:
660   .ascii  "Print this information and quit.\n\n\000"
661   .align  2
662 $LC21:
663   .ascii  "Examples:\n\000"
664   .align  2
665 $LC22:
666   .ascii  "  tp0 < in.txt > out.txt\n\000"
667   .align  2
668 $LC23:
669   .ascii  "  cat in.txt | tp0 > out.txt\n\n\000"
670   .text
671   .align  2
672   .globl  printHelp
673   .ent  printHelp
674 printHelp:
675   .frame  $fp,40,$31    # vars= 0, regs= 3/0, args= 16, extra= 8
676   .mask 0xd0000000,-8
677   .fmask  0x00000000,0
678   .set  noreorder
679   .cpload $25
680   .set  reorder
```

```
681     subu   $sp,$sp,40
682     .cprestore 16
683     sw   $31,32($sp)
684     sw   $fp,28($sp)
685     sw   $28,24($sp)
686     move   $fp,$sp
687     la   $4,__sF+88
688     la   $5,$LC12
689     la   $25,fprintf
690     jal  $31,$25
691     la   $4,__sF+88
692     la   $5,$LC13
693     la   $25,fprintf
694     jal  $31,$25
695     la   $4,__sF+88
696     la   $5,$LC14
697     la   $25,fprintf
698     jal  $31,$25
699     la   $4,__sF+88
700     la   $5,$LC15
701     la   $25,fprintf
702     jal  $31,$25
703     la   $4,__sF+88
704     la   $5,$LC16
705     la   $25,fprintf
706     jal  $31,$25
707     la   $4,__sF+88
708     la   $5,$LC17
709     la   $25,fprintf
710     jal  $31,$25
711     la   $4,__sF+88
712     la   $5,$LC18
713     la   $25,fprintf
714     jal  $31,$25
715     la   $4,__sF+88
716     la   $5,$LC19
717     la   $25,fprintf
718     jal  $31,$25
719     la   $4,__sF+88
720     la   $5,$LC20
721     la   $25,fprintf
722     jal  $31,$25
723     la   $4,__sF+88
724     la   $5,$LC21
725     la   $25,fprintf
726     jal  $31,$25
727     la   $4,__sF+88
728     la   $5,$LC22
729     la   $25,fprintf
730     jal  $31,$25
731     la   $4,__sF+88
732     la   $5,$LC23
733     la   $25,fprintf
734     jal  $31,$25
735     move   $sp,$fp
736     lw   $31,32($sp)
737     lw   $fp,28($sp)
738     addu   $sp,$sp,40
739     j  $31
740     .end   printHelp
741     .size printHelp, .-printHelp
742     .rdata
```

```
743   .align  2
744 $LC24:
745   .ascii  "Copyright (c) 2016\n\000"
746   .align  2
747 $LC25:
748   .ascii  "MIPS - Infraestructura b\303\241sica. v1.0.0\n\n\000"
749   .text
750   .align  2
751   .globl  printVersion
752   .ent  printVersion
753 printVersion:
754   .frame  $fp,40,$31    # vars= 0, regs= 3/0, args= 16, extra= 8
755   .mask 0xd0000000,-8
756   .fmask  0x00000000,0
757   .set  noreorder
758   .cpload $25
759   .set  reorder
760   subu  $sp,$sp,40
761   .cprestore 16
762   sw  $31,32($sp)
763   sw  $fp,28($sp)
764   sw  $28,24($sp)
765   move  $fp,$sp
766   la  $4,__sF+88
767   la  $5,$LC24
768   la  $25,fprintf
769   jal $31,$25
770   la  $4,__sF+88
771   la  $5,$LC25
772   la  $25,fprintf
773   jal $31,$25
774   move  $sp,$fp
775   lw  $31,32($sp)
776   lw  $fp,28($sp)
777   addu  $sp,$sp,40
778   j $31
779   .end  printVersion
780   .size printVersion, .-printVersion
781   .align  2
782   .globl  matrices_multiply
783   .ent  matrices_multiply
784 matrices_multiply:
785   .frame  $fp,96,$31    # vars= 56, regs= 3/0, args= 16, extra= 8
786   .mask 0xd0000000,-8
787   .fmask  0x00000000,0
788   .set  noreorder
789   .cpload $25
790   .set  reorder
791   subu  $sp,$sp,96
792   .cprestore 16
793   sw  $31,88($sp)
794   sw  $fp,84($sp)
795   sw  $28,80($sp)
796   move  $fp,$sp
797   sw  $4,96($fp)
798   sw  $5,100($fp)
799   li  $4,1      # 0x1
800   la  $25,malloc
801   jal $31,$25
802   sw  $2,28($fp)
803   sw  $0,32($fp)
804   sw  $0,36($fp)
```

```
805     sw   $0,40($fp)
806     sw   $0,44($fp)
807     sw   $0,48($fp)
808   $L62:
809     lw   $3,96($fp)
810     lw   $2,96($fp)
811     lw   $2,4($2)
812     addu   $2,$2,-1
813     sw   $2,4($3)
814     bgez   $2,$L65
815     lw   $4,96($fp)
816     la   $25,__srget
817     jal  $31,$25
818     sb   $2,76($fp)
819     b $L66
820   $L65:
821     lw   $2,96($fp)
822     lw   $3,0($2)
823     move   $4,$3
824     lbu  $4,0($4)
825     sb   $4,76($fp)
826     addu   $3,$3,1
827     sw   $3,0($2)
828   $L66:
829     lbu  $2,76($fp)
830     sb   $2,24($fp)
831     sll  $2,$2,24
832     sra  $3,$2,24
833     li   $2,-1       # 0xffffffffffffffff
834     bne  $3,$2,$L64
835     b $L63
836   $L64:
837     lw   $2,32($fp)
838     addu   $2,$2,1
839     sw   $2,32($fp)
840     lw   $4,28($fp)
841     lw   $5,32($fp)
842     la   $25,realloc
843     jal  $31,$25
844     sw   $2,28($fp)
845     lw   $3,28($fp)
846     lw   $2,32($fp)
847     addu   $2,$3,$2
848     addu   $3,$2,-1
849     lbu  $2,24($fp)
850     sb   $2,0($3)
851     lb   $3,24($fp)
852     li   $2,10       # 0xa
853     bne  $3,$2,$L62
854     addu   $2,$fp,36
855     lw   $4,28($fp)
856     lw   $5,32($fp)
857     move   $6,$2
858     la   $25,read_arguments
859     jal  $31,$25
860     sw   $2,52($fp)
861     lw   $4,36($fp)
862     la   $25,get_amount_element
863     jal  $31,$25
864     sw   $2,40($fp)
865     lw   $4,36($fp)
866     la   $25,get_matrix_elements
```

```
867     jal $31,$25
868     sw  $2,44($fp)
869     lw  $2,52($fp)
870     sw  $2,56($fp)
871     lw  $2,44($fp)
872     sll $3,$2,3
873     lw  $2,52($fp)
874     addu  $2,$2,$3
875     sw  $2,60($fp)
876     lw  $4,36($fp)
877     lw  $5,36($fp)
878     la  $25,create_matrix
879     jal $31,$25
880     sw  $2,64($fp)
881     lw  $3,64($fp)
882     lw  $2,56($fp)
883     sw  $2,8($3)
884     lw  $4,36($fp)
885     lw  $5,36($fp)
886     la  $25,create_matrix
887     jal $31,$25
888     sw  $2,68($fp)
889     lw  $3,68($fp)
890     lw  $2,60($fp)
891     sw  $2,8($3)
892     lw  $4,64($fp)
893     lw  $5,68($fp)
894     la  $25,matrix_multiply
895     jal $31,$25
896     sw  $2,72($fp)
897     lw  $4,100($fp)
898     lw  $5,72($fp)
899     la  $25,print_matrix
900     jal $31,$25
901     sw  $2,48($fp)
902     lw  $4,64($fp)
903     la  $25,destroy_matrix
904     jal $31,$25
905     lw  $4,68($fp)
906     la  $25,destroy_matrix
907     jal $31,$25
908     lw  $4,72($fp)
909     la  $25,destroy_matrix
910     jal $31,$25
911     lw  $4,52($fp)
912     la  $25,free
913     jal $31,$25
914     sw  $0,32($fp)
915     b $L62
916 $L63:
917     lw  $4,28($fp)
918     la  $25,free
919     jal $31,$25
920     lw  $2,48($fp)
921     move  $sp,$fp
922     lw  $31,88($sp)
923     lw  $fp,84($sp)
924     addu  $sp,$sp,96
925     j $31
926     .end  matrices_multiply
927     .size matrices_multiply, .-matrices_multiply
928     .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

../src/mips/main.s

6.3. Bibliografía

Referencias

[1] *GXemul*
`http://gavare.se/gxemul/`

[2] *The NetBSD project*
`http://www.netbsd.org/`

[3] *Row-major order (Wikipedia)*
`https://en.wikipedia.org/wiki/Row-major_order`