U.B.A. FACULTAD DE INGENIERÍA

Departamento de Electrónica
Organización de Computadoras 66-20
Ing. Informática

Trabajo práctico N°0
Infraestructura básica

Apellido y Nombre:	*Padrón:*
Cabrera, Jorge	93310
Capolupo, Mauro	90283
Serra, Diego Adrián	92354

Fecha de Entrega : 05/04/2016
Fecha de Aprobación :
Calificación :
Firma de Aprobación :

1. Diseño e implementación del programa

El programa permite obtener el producto de dos matrices cuadradas a partir de un stream de datos. Cada parte del flujo de datos mantiene una misma estructura, el primer caracter corresponde a la dimensión de las matrices, y los subsiguientes tokens corresponde a los valores.

Por cada linea a procesar se van a esperar n * n * 2 números décimales, siendo n el primer caracter leído, correspondiente a las dimensiones de las matrices. Si el número es distinto de este valor, o alguno de los caracteres no corresponde a un número décimal, entonces se produce un error.

2. Comando(s) para compilar el programa

gcc -Wall -O0 -o Tp0.x Tp0.c

3. Pruebas

3.1. Prueba 1

El archivo prueba.txt es
2 1 2 3 4 5 6 7 8
3 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
respresenta las operaciones

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \times \begin{pmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{pmatrix} = \begin{pmatrix} 84 & 90 & 96 \\ 201 & 216 & 231 \\ 318 & 342 & 366 \end{pmatrix}$$

por salida estandar se vera
2 19 22 43 50
3 84 90 96 201 216 231 318 342 366

3.2. Prueba 2

El archivo prueba.txt es
2 1 2 3 4 5 6 7

respresenta la operacion

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 & 6 \\ 7 & - \end{pmatrix}$$

por salida estandar se vera
Cantidad incorrecta de parametros para matriz de dimension 2.

3.3. Prueba 3

El archivo de prueba.txt es
2 a 2 3 4 5 6 7 8
respresenta las operaciones

$$\begin{pmatrix} a & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

por salida estandar se vera
Lectura de caracter no valido.
Cantidad incorrecta de parametros para matriz de dimension 2.

4. Codigo fuente, en lenguaje C

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6  int cantProcesos = 1;
7
8  typedef struct matrix {
9     size_t rows;
10    size_t cols;
11    float *array;
12 } matrix_t;
13
14 // Constructor de matrix_t
15 matrix_t* create_matrix(size_t rows, size_t cols) {
16    matrix_t* matrix = malloc(sizeof(matrix_t));
17    if(matrix == NULL){
18        return NULL;
19    }
20    matrix->rows = rows;
21    matrix->cols = cols;
22    matrix->array = (float*)calloc(rows * cols, sizeof(float));
23    if (matrix->array == NULL){
24       return NULL;
25    }
26    return matrix;
27 }
28
29 // Destructor de matrix_t
30 void destroy_matrix(matrix_t* m) {
31    free(m->array);
32    m->array = NULL;
33    free(m);
34    m=NULL;
35 }
36
37 // Imprime matrix_t sobre el file pointer fp en el formato ↩
      solicitado
38 // por el enunciado
39 int print_matrix(FILE* fp, matrix_t* m) {
40    int i = 0;
```

```
41    fprintf(fp, "%d ", (int) (m->cols));
42    while (i < (m->cols) * (m->cols)) {
43       fprintf(fp, "%f ", m->array[i]);
44       i++;
45    }
46    fprintf(fp, "\n");
47    return 0;
48 }
49
50 // Multiplica las matrices en m1 y m2
51 matrix_t* matrix_multiply(matrix_t* m1, matrix_t* m2) {
52    int m1_index = 0;
53    int m2_index = 0;
54    int m2_aux = 0;
55    int index = 0;
56    matrix_t* result = create_matrix(m1->rows, m1->cols);
57
58    for (m1_index = 0; m1_index <= m1->rows * m1->cols;) {
59       m1_index = (index / m1->cols) * m1->rows;
60       result->array[index]=0;
61       for (m2_aux = 0; m2_aux < m2->rows;) {
62          result->array[index] += m1->array[m1_index] * m2->↩
              array[m2_index];
63          m2_aux++;
64          m1_index++;
65          m2_index += m2->rows;
66       }
67       index++;
68       m2_index = index %m2->cols;
69       m2_aux = 0;
70    }
71    return result;
72 }
73
74 void show_help(){
75    printf("Usage:\n");
76    printf("\t tp0 -h \n");
77    printf("\t tp0 -V \n");
78    printf("\t tp0 < in_file > out_file \n");
79    printf("Options:\n");
80    printf("\t -V, --version \t Print version and quit. \n");
81    printf("\t -h, --help \t Print this information and quit. \n↩
          ");
82    printf("Examples:\n");
83    printf("\t tp0 < in.txt > out.txt \n");
```

```
84    printf("\t cat in.txt | tp0 > out.txt \n");
85 }
86
87 void show_version(){
88    printf("version xx \n");
89 }
90
91
92 int leerTamanio(int* cont, int* err) {
93    int n=0;
94    int resp;
95    resp = scanf("%d", &n);
96
97    if (resp == EOF){
98       *cont=0;
99       return 0;
100   }else if (n <= 0){
101      *err = 1;
102      fprintf(stderr, "no se pudo obtener ñtamaio de matrix. ↩
         Fila: %d\n",cantProcesos);
103      return 0;
104   };
105   return n;
106 }
107
108 char* readString(int* errRead) {
109    char c;
110    char *string;
111    int continuar = 1;
112    c = getchar();
113    while (c == 32){ // blank
114      c = getchar();
115    }
116    if ((c>=48 && c<=57) || (c>=45 && c<=46) || (c==43) || (c↩
       ==101)){ //numeros | - | . | + | e |
117      char aux[1];
118      aux[0] = c;
119      aux[1] = '\0';
120      string = (char*) malloc((strlen(aux)+1)*sizeof(char));
121      strcpy(string, aux);
122    }else if(c == 10){ // newline
123      return NULL;
124    }else{
125      *errRead = 1;
```

```
126       fprintf(stderr,"Lectura de caracter no valido. Linea: %d\n",cantProcesos);
127       return NULL;
128     };
129     do{
130         c = getchar();
131         if ((c>=48 && c<=57) || (c>=45 && c<=46) || (c==43) || (c==101)){
132          char aux[1];
133          aux[0] = c;
134          aux[1] = '\0';
135             string = (char*)realloc(string, (strlen(string)+strlen(aux)+1)*sizeof(char));
136             strcat(string, aux);
137         }else if ((c == 32) || (c == 10)){
138          continuar = 0;
139         }else{
140          *errRead = 1;
141          fprintf(stderr,"Lectura de caracter no valido. Linea: %d\n",cantProcesos);
142          free(string);
143          return NULL;
144         };
145     }while(continuar);
146     return string;
147 }
148
149
150 void fillMatrix(int tam, matrix_t *matrix, int* err) {
151          char *token;
152          int i = 0;
153          int validNumber = 0;
154          int errFill = 0;
155             token = readString(&errFill);
156             while ((token != NULL) && (i < (tam*tam)) && !errFill){
157                float d;
158                validNumber = 0;
159                validNumber = sscanf(token, "%g", &d);
160                if (validNumber > 0 ) {
161                   matrix->array[i]=d;
162                   i++;
163                }else{
164                   fprintf(stderr,"Numero con formato incorrecto. Linea: %d\n", cantProcesos);
```

```
165                     errFill = 1;
166                 }
167                 free(token);
168                 if (i != (tam*tam)){
169                     token = readString(&errFill);
170                 }
171             }
172             if ((token == NULL) && (i < (tam*tam))) {
173                 fprintf(stderr,"Cantidad incorrecta de ↩
                        parametros para matriz de dimension %d. ↩
                        Linea: %d\n", tam, cantProcesos);
174                 errFill = 1;
175             }
176             if (errFill){ *err = errFill; };
177 }
178
179 int main(int argc, char **argv) {
180
181     if (argc > 1) {
182         if ((strcmp(argv[1], "-h") == 0) || (strcmp(argv[1], "--↩
                help") == 0)) {
183             show_help();
184             return 0;
185         } else if ((strcmp(argv[1], "-V") == 0)
186                 || (strcmp(argv[1], "--version") == 0)) {
187             show_version();
188             return 0;
189         }else{
190             printf("Paramatro incorrecto. Ingrese -h para ayuda.\n↩
                    ");
191             return 0;
192         };
193     }
194
195     matrix_t* matrix_a=NULL;
196     matrix_t* matrix_b=NULL;
197     matrix_t* matrix_c=NULL;
198     int continuar = 1;
199     int err = 0;
200     size_t n = leerTamanio(&continuar,&err);
201     while(continuar && !err){
202         matrix_a = create_matrix(n,n);
203         matrix_b = create_matrix(n,n);
204         fillMatrix(n,matrix_a, &err);
205         if (!err){
```

```
206             fillMatrix(n,matrix_b, &err);
207         }
208         if (!err){
209             matrix_c = matrix_multiply(matrix_a,matrix_b);
210             print_matrix(stdout, matrix_c);
211             if (matrix_c != NULL) { destroy_matrix(matrix_c); };
212         }
213         if (matrix_a != NULL) { destroy_matrix(matrix_a); };
214         if (matrix_b != NULL) { destroy_matrix(matrix_b); };
215 
216         cantProcesos++;
217         if (!err){
218             n = leerTamanio(&continuar, &err);
219         }
220     }
221     return EXIT_SUCCESS;
222 }
```

5. Codigo MIPS32 generado por el compilador

```
.file 1 "Tp0.c"
.section .mdebug.abi32
.previous
.abicalls
.globl cantProcesos
.data
.align 2
.type cantProcesos, @object
.size cantProcesos, 4
cantProcesos:
.word 1
.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 $v1,24($fp)
lw $v0,48($fp)
sw $v0,0($v1)
lw $v1,24($fp)
lw $v0,52($fp)
sw $v0,4($v1)
```

```
lw $s0,24($fp)
lw $v1,48($fp)
lw $v0,52($fp)
mult $v1,$v0
mflo $v0
sll $v0,$v0,3
addu $v0,$v0,4
move $a0,$v0
la $t9,malloc
jal $ra,$t9
sw $v0,8($s0)
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)
lw $a0,8($v0)
la $t9,free
jal $ra,$t9
lw $v0,40($fp)
sw $zero,8($v0)
lw $a0,40($fp)
la $t9,free
```

```
jal $ra,$t9
sw $zero,40($fp)
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
.rdata
.align 2
$LC0:
.ascii "%d \000"
.align 2
$LC1:
.ascii "%f \000"
.align 2
$LC2:
.ascii "\n\000"
.text
.align 2
.globl print_matrix
.ent print_matrix
print_matrix:
.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)
lw $v0,52($fp)
lw $a0,48($fp)
la $a1,$LC0
lw $a2,4($v0)
la $t9,fprintf
jal $ra,$t9
```

```
$L20:
lw $v0,52($fp)
lw $v1,52($fp)
lw $a0,4($v0)
lw $v0,4($v1)
mult $a0,$v0
mflo $v1
lw $v0,24($fp)
sltu $v0,$v0,$v1
bne $v0,$zero,$L22
b $L21
$L22:
lw $a0,52($fp)
lw $v0,24($fp)
sll $v1,$v0,3
lw $v0,8($a0)
addu $v0,$v1,$v0
lw $a0,48($fp)
la $a1,$LC1
lw $a2,0($v0)
lw $a3,4($v0)
la $t9,fprintf
jal $ra,$t9
lw $v0,24($fp)
addu $v0,$v0,1
sw $v0,24($fp)
b $L20
$L21:
lw $a0,48($fp)
la $a1,$LC2
la $t9,fprintf
jal $ra,$t9
move $v0,$zero
move $sp,$fp
lw $ra,40($sp)
lw $fp,36($sp)
addu $sp,$sp,48
j $ra
.end print_matrix
.size print_matrix, .-print_matrix
.align 2
.globl matrix_multiply
.ent matrix_multiply
matrix_multiply:
.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)
sw $zero,24($fp)
sw $zero,28($fp)
sw $zero,32($fp)
sw $zero,36($fp)
lw $v0,64($fp)
lw $v1,64($fp)
lw $a0,0($v0)
lw $a1,4($v1)
la $t9,create_matrix
jal $ra,$t9
sw $v0,40($fp)
sw $zero,24($fp)
$L24:
lw $v0,64($fp)
lw $v1,64($fp)
lw $a0,0($v0)
lw $v0,4($v1)
mult $a0,$v0
mflo $v1
lw $v0,24($fp)
sltu $v0,$v1,$v0
beq $v0,$zero,$L27
b $L25
$L27:
lw $v0,64($fp)
lw $v1,36($fp)
lw $v0,4($v0)
divu $0,$v1,$v0
mflo $v1
.set noreorder
bne $v0,$0,1f
nop
```

```
break 7
1:
.set reorder
lw $v0,64($fp)
lw $v0,0($v0)
mult $v1,$v0
mflo $v0
sw $v0,24($fp)
lw $a0,40($fp)
lw $v0,36($fp)
sll $v1,$v0,3
lw $v0,8($a0)
addu $v0,$v1,$v0
sw $zero,0($v0)
sw $zero,4($v0)
sw $zero,32($fp)
$L28:
lw $v0,68($fp)
lw $v1,32($fp)
lw $v0,0($v0)
sltu $v0,$v1,$v0
bne $v0,$zero,$L31
b $L29
$L31:
lw $a0,40($fp)
lw $v0,36($fp)
sll $v1,$v0,3
lw $v0,8($a0)
addu $a3,$v1,$v0
lw $a0,40($fp)
lw $v0,36($fp)
sll $v1,$v0,3
lw $v0,8($a0)
addu $a2,$v1,$v0
lw $a0,64($fp)
lw $v0,24($fp)
sll $v1,$v0,3
lw $v0,8($a0)
addu $a1,$v1,$v0
lw $a0,68($fp)
lw $v0,28($fp)
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 $f0,0($a2)
add.d $f0,$f0,$f2
s.d $f0,0($a3)
lw $v0,32($fp)
addu $v0,$v0,1
sw $v0,32($fp)
lw $v0,24($fp)
addu $v0,$v0,1
sw $v0,24($fp)
lw $v0,68($fp)
lw $v1,28($fp)
lw $v0,0($v0)
addu $v0,$v1,$v0
sw $v0,28($fp)
b $L28
$L29:
lw $v0,36($fp)
addu $v0,$v0,1
sw $v0,36($fp)
lw $v0,68($fp)
lw $v1,36($fp)
lw $v0,4($v0)
divu $0,$v1,$v0
mfhi $v1
.set noreorder
bne $v0,$0,1f
nop
break 7
1:
.set reorder
sw $v1,28($fp)
sw $zero,32($fp)
b $L24
$L25:
lw $v0,40($fp)
move $sp,$fp
lw $ra,56($sp)
lw $fp,52($sp)
addu $sp,$sp,64
j $ra
.end matrix_multiply
.size matrix_multiply, .-matrix_multiply
.rdata
```

```
.align 2
$LC3:
.ascii "Usage:\n\000"
.align 2
$LC4:
.ascii "\t tp0 -h \n\000"
.align 2
$LC5:
.ascii "\t tp0 -V \n\000"
.align 2
$LC6:
.ascii "\t tp0 < in_file > out_file \n\000"
.align 2
$LC7:
.ascii "Options:\n\000"
.align 2
$LC8:
.ascii "\t -V, --version \t Print version and quit. \n\000"
.align 2
$LC9:
.ascii "\t -h, --help \t Print this information and quit. \n\000"
.align 2
$LC10:
.ascii "Examples:\n\000"
.align 2
$LC11:
.ascii "\t tp0 < in.txt > out.txt \n\000"
.align 2
$LC12:
.ascii "\t cat in.txt | tp0 > out.txt \n\000"
.text
.align 2
.globl show_help
.ent show_help
show_help:
.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
la $a0,$LC3
la $t9,printf
jal $ra,$t9
la $a0,$LC4
la $t9,printf
jal $ra,$t9
la $a0,$LC5
la $t9,printf
jal $ra,$t9
la $a0,$LC6
la $t9,printf
jal $ra,$t9
la $a0,$LC7
la $t9,printf
jal $ra,$t9
la $a0,$LC8
la $t9,printf
jal $ra,$t9
la $a0,$LC9
la $t9,printf
jal $ra,$t9
la $a0,$LC10
la $t9,printf
jal $ra,$t9
la $a0,$LC11
la $t9,printf
jal $ra,$t9
la $a0,$LC12
la $t9,printf
jal $ra,$t9
move $sp,$fp
lw $ra,32($sp)
lw $fp,28($sp)
addu $sp,$sp,40
j $ra
.end show_help
.size show_help, .-show_help
.rdata
.align 2
$LC13:
.ascii "version xx \n\000"
.text
.align 2
```

```
.globl show_version
.ent show_version
show_version:
.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
la $a0,$LC13
la $t9,printf
jal $ra,$t9
move $sp,$fp
lw $ra,32($sp)
lw $fp,28($sp)
addu $sp,$sp,40
j $ra
.end show_version
.size show_version, .-show_version
.rdata
.align 2
$LC14:
.ascii "%d\000"
.align 2
$LC15:
.ascii "no se pudo obtener tama\303\261io de matrix. Fila: %d\n\000"
.text
.align 2
.globl leerTamanio
.ent leerTamanio
leerTamanio:
.frame $fp,56,$ra # vars= 16, regs= 3/0, args= 16, extra= 8
.mask 0xd0000000,-8
.fmask 0x00000000,0
.set noreorder
.cpload $t9
.set reorder
subu $sp,$sp,56
.cprestore 16
```

```
sw $ra,48($sp)
sw $fp,44($sp)
sw $gp,40($sp)
move $fp,$sp
sw $a0,56($fp)
sw $a1,60($fp)
sw $zero,24($fp)
la $a0,$LC14
addu $a1,$fp,24
la $t9,scanf
jal $ra,$t9
sw $v0,28($fp)
lw $v1,28($fp)
li $v0,-1 # 0xffffffffffffffff
bne $v1,$v0,$L35
lw $v0,56($fp)
sw $zero,0($v0)
sw $zero,32($fp)
b $L34
$L35:
lw $v0,24($fp)
bgtz $v0,$L36
lw $v1,60($fp)
li $v0,1 # 0x1
sw $v0,0($v1)
la $a0,__sF+176
la $a1,$LC15
lw $a2,cantProcesos
la $t9,fprintf
jal $ra,$t9
sw $zero,32($fp)
b $L34
$L36:
lw $v0,24($fp)
sw $v0,32($fp)
$L34:
lw $v0,32($fp)
move $sp,$fp
lw $ra,48($sp)
lw $fp,44($sp)
addu $sp,$sp,56
j $ra
.end leerTamanio
.size leerTamanio, .-leerTamanio
.rdata
```

```
.align 2
$LC16:
.ascii "Lectura de caracter no valido. Linea: %d\n\000"
.text
.align 2
.globl readString
.ent readString
readString:
.frame $fp,64,$ra # vars= 24, regs= 4/0, args= 16, extra= 8
.mask 0xd0010000,-4
.fmask 0x00000000,0
.set noreorder
.cpload $t9
.set reorder
subu $sp,$sp,64
.cprestore 16
sw $ra,60($sp)
sw $fp,56($sp)
sw $gp,52($sp)
sw $s0,48($sp)
move $fp,$sp
sw $a0,64($fp)
li $v0,1 # 0x1
sw $v0,32($fp)
lw $v0,__sF+4
addu $v0,$v0,-1
sw $v0,__sF+4
bgez $v0,$L39
la $a0,__sF
la $t9,__srget
jal $ra,$t9
sb $v0,44($fp)
b $L40
$L39:
la $v0,__sF
lw $v1,0($v0)
move $a0,$v1
lbu $a0,0($a0)
sb $a0,44($fp)
addu $v1,$v1,1
sw $v1,0($v0)
$L40:
lbu $v0,44($fp)
sb $v0,24($fp)
$L41:
```

```
lb $v1,24($fp)
li $v0,32 # 0x20
beq $v1,$v0,$L43
b $L42
$L43:
lw $v0,__sF+4
addu $v0,$v0,-1
sw $v0,__sF+4
bgez $v0,$L44
la $a0,__sF
la $t9,__srget
jal $ra,$t9
sb $v0,45($fp)
b $L45
$L44:
la $v0,__sF
lw $v1,0($v0)
move $a0,$v1
lbu $a0,0($a0)
sb $a0,45($fp)
addu $v1,$v1,1
sw $v1,0($v0)
$L45:
lbu $v0,45($fp)
sb $v0,24($fp)
b $L41
$L42:
lb $v0,24($fp)
slt $v0,$v0,48
bne $v0,$zero,$L48
lb $v0,24($fp)
slt $v0,$v0,58
bne $v0,$zero,$L47
$L48:
lb $v0,24($fp)
slt $v0,$v0,45
bne $v0,$zero,$L49
lb $v0,24($fp)
slt $v0,$v0,47
bne $v0,$zero,$L47
$L49:
lb $v1,24($fp)
li $v0,43 # 0x2b
beq $v1,$v0,$L47
lb $v1,24($fp)
```

```
li $v0,101 # 0x65
beq $v1,$v0,$L47
b $L46
$L47:
lbu $v0,24($fp)
sb $v0,36($fp)
sb $zero,37($fp)
addu $v0,$fp,36
move $a0,$v0
la $t9,strlen
jal $ra,$t9
addu $v0,$v0,1
move $a0,$v0
la $t9,malloc
jal $ra,$t9
sw $v0,28($fp)
addu $v0,$fp,36
lw $a0,28($fp)
move $a1,$v0
la $t9,strcpy
jal $ra,$t9
b $L50
$L46:
lb $v1,24($fp)
li $v0,10 # 0xa
bne $v1,$v0,$L51
sw $zero,40($fp)
b $L38
$L51:
lw $v1,64($fp)
li $v0,1 # 0x1
sw $v0,0($v1)
la $a0,__sF+176
la $a1,$LC16
lw $a2,cantProcesos
la $t9,fprintf
jal $ra,$t9
sw $zero,40($fp)
b $L38
$L50:
.set noreorder
nop
.set reorder
$L53:
lw $v0,__sF+4
```

```
addu $v0,$v0,-1
sw $v0,__sF+4
bgez $v0,$L56
la $a0,__sF
la $t9,__srget
jal $ra,$t9
sb $v0,46($fp)
b $L57
$L56:
la $v0,__sF
lw $v1,0($v0)
move $a0,$v1
lbu $a0,0($a0)
sb $a0,46($fp)
addu $v1,$v1,1
sw $v1,0($v0)
$L57:
lbu $v0,46($fp)
sb $v0,24($fp)
lb $v0,24($fp)
slt $v0,$v0,48
bne $v0,$zero,$L60
lb $v0,24($fp)
slt $v0,$v0,58
bne $v0,$zero,$L59
$L60:
lb $v0,24($fp)
slt $v0,$v0,45
bne $v0,$zero,$L61
lb $v0,24($fp)
slt $v0,$v0,47
bne $v0,$zero,$L59
$L61:
lb $v1,24($fp)
li $v0,43 # 0x2b
beq $v1,$v0,$L59
lb $v1,24($fp)
li $v0,101 # 0x65
beq $v1,$v0,$L59
b $L58
$L59:
lbu $v0,24($fp)
sb $v0,37($fp)
sb $zero,38($fp)
lw $a0,28($fp)
```

```
la $t9,strlen
jal $ra,$t9
move $s0,$v0
addu $v0,$fp,37
move $a0,$v0
la $t9,strlen
jal $ra,$t9
addu $v0,$s0,$v0
addu $v0,$v0,1
lw $a0,28($fp)
move $a1,$v0
la $t9,realloc
jal $ra,$t9
sw $v0,28($fp)
addu $v0,$fp,37
lw $a0,28($fp)
move $a1,$v0
la $t9,strcat
jal $ra,$t9
b $L55
$L58:
lb $v1,24($fp)
li $v0,32 # 0x20
beq $v1,$v0,$L64
lb $v1,24($fp)
li $v0,10 # 0xa
beq $v1,$v0,$L64
b $L63
$L64:
sw $zero,32($fp)
b $L55
$L63:
lw $v1,64($fp)
li $v0,1 # 0x1
sw $v0,0($v1)
la $a0,__sF+176
la $a1,$LC16
lw $a2,cantProcesos
la $t9,fprintf
jal $ra,$t9
lw $a0,28($fp)
la $t9,free
jal $ra,$t9
sw $zero,40($fp)
b $L38
```

```
$L55:
lw $v0,32($fp)
bne $v0,$zero,$L53
lw $v0,28($fp)
sw $v0,40($fp)
$L38:
lw $v0,40($fp)
move $sp,$fp
lw $ra,60($sp)
lw $fp,56($sp)
lw $s0,48($sp)
addu $sp,$sp,64
j $ra
.end readString
.size readString, .-readString
.rdata
.align 2
$LC17:
.ascii "%g\000"
.align 2
$LC18:
.ascii "Numero con formato incorrecto. Linea: %d\n\000"
.align 2
$LC19:
.ascii "Cantidad incorrecta de parametros para matriz de dimensi"
.ascii "on %d. Linea: %d\n\000"
.text
.align 2
.globl fillMatrix
.ent fillMatrix
fillMatrix:
.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)
```

```
sw $a2,72($fp)
sw $zero,28($fp)
sw $zero,32($fp)
sw $zero,36($fp)
addu $v0,$fp,36
move $a0,$v0
la $t9,readString
jal $ra,$t9
sw $v0,24($fp)
$L68:
lw $v0,24($fp)
beq $v0,$zero,$L69
lw $v1,64($fp)
lw $v0,64($fp)
mult $v1,$v0
mflo $v1
lw $v0,28($fp)
slt $v0,$v0,$v1
beq $v0,$zero,$L69
lw $v0,36($fp)
bne $v0,$zero,$L69
sw $zero,32($fp)
addu $v0,$fp,40
lw $a0,24($fp)
la $a1,$LC17
move $a2,$v0
la $t9,sscanf
jal $ra,$t9
sw $v0,32($fp)
lw $v0,32($fp)
blez $v0,$L72
lw $a0,68($fp)
lw $v0,28($fp)
sll $v1,$v0,3
lw $v0,8($a0)
addu $v0,$v1,$v0
l.s $f0,40($fp)
cvt.d.s $f0,$f0
s.d $f0,0($v0)
lw $v0,28($fp)
addu $v0,$v0,1
sw $v0,28($fp)
b $L73
$L72:
la $a0,__sF+176
```

```
la $a1,$LC18
lw $a2,cantProcesos
la $t9,fprintf
jal $ra,$t9
li $v0,1 # 0x1
sw $v0,36($fp)
$L73:
lw $a0,24($fp)
la $t9,free
jal $ra,$t9
lw $v1,64($fp)
lw $v0,64($fp)
mult $v1,$v0
mflo $v1
lw $v0,28($fp)
beq $v0,$v1,$L68
addu $v0,$fp,36
move $a0,$v0
la $t9,readString
jal $ra,$t9
sw $v0,24($fp)
b $L68
$L69:
lw $v0,24($fp)
bne $v0,$zero,$L75
lw $v1,64($fp)
lw $v0,64($fp)
mult $v1,$v0
mflo $v1
lw $v0,28($fp)
slt $v0,$v0,$v1
beq $v0,$zero,$L75
la $a0,__sF+176
la $a1,$LC19
lw $a2,64($fp)
lw $a3,cantProcesos
la $t9,fprintf
jal $ra,$t9
li $v0,1 # 0x1
sw $v0,36($fp)
$L75:
lw $v0,36($fp)
beq $v0,$zero,$L67
lw $v1,72($fp)
lw $v0,36($fp)
```

```
sw $v0,0($v1)
$L67:
move $sp,$fp
lw $ra,56($sp)
lw $fp,52($sp)
addu $sp,$sp,64
j $ra
.end fillMatrix
.size fillMatrix, .-fillMatrix
.rdata
.align 2
$LC20:
.ascii "-h\000"
.align 2
$LC21:
.ascii "--help\000"
.align 2
$LC22:
.ascii "-V\000"
.align 2
$LC23:
.ascii "--version\000"
.align 2
$LC24:
.ascii "Paramatro incorrecto. Ingrese -h para ayuda.\n\000"
.text
.align 2
.globl main
.ent main
main:
.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)
```

```
slt $v0,$v0,2
bne $v0,$zero,$L78
lw $v0,76($fp)
addu $v0,$v0,4
lw $a0,0($v0)
la $a1,$LC20
la $t9,strcmp
jal $ra,$t9
beq $v0,$zero,$L80
lw $v0,76($fp)
addu $v0,$v0,4
lw $a0,0($v0)
la $a1,$LC21
la $t9,strcmp
jal $ra,$t9
bne $v0,$zero,$L79
$L80:
la $t9,show_help
jal $ra,$t9
sw $zero,48($fp)
b $L77
$L79:
lw $v0,76($fp)
addu $v0,$v0,4
lw $a0,0($v0)
la $a1,$LC22
la $t9,strcmp
jal $ra,$t9
beq $v0,$zero,$L83
lw $v0,76($fp)
addu $v0,$v0,4
lw $a0,0($v0)
la $a1,$LC23
la $t9,strcmp
jal $ra,$t9
bne $v0,$zero,$L82
$L83:
la $t9,show_version
jal $ra,$t9
sw $zero,48($fp)
b $L77
$L82:
la $a0,$LC24
la $t9,printf
jal $ra,$t9
```

```
sw $zero,48($fp)
b $L77
$L78:
sw $zero,24($fp)
sw $zero,28($fp)
sw $zero,32($fp)
li $v0,1 # 0x1
sw $v0,36($fp)
sw $zero,40($fp)
addu $v0,$fp,36
addu $v1,$fp,40
move $a0,$v0
move $a1,$v1
la $t9,leerTamanio
jal $ra,$t9
sw $v0,44($fp)
$L85:
lw $v0,36($fp)
beq $v0,$zero,$L86
lw $v0,40($fp)
bne $v0,$zero,$L86
lw $a0,44($fp)
lw $a1,44($fp)
la $t9,create_matrix
jal $ra,$t9
sw $v0,24($fp)
lw $a0,44($fp)
lw $a1,44($fp)
la $t9,create_matrix
jal $ra,$t9
sw $v0,28($fp)
addu $v0,$fp,40
lw $a0,44($fp)
lw $a1,24($fp)
move $a2,$v0
la $t9,fillMatrix
jal $ra,$t9
lw $v0,40($fp)
bne $v0,$zero,$L89
addu $v0,$fp,40
lw $a0,44($fp)
lw $a1,28($fp)
move $a2,$v0
la $t9,fillMatrix
jal $ra,$t9
```

```
$L89:
lw $v0,40($fp)
bne $v0,$zero,$L90
lw $a0,24($fp)
lw $a1,28($fp)
la $t9,matrix_multiply
jal $ra,$t9
sw $v0,32($fp)
la $a0,__sF+88
lw $a1,32($fp)
la $t9,print_matrix
jal $ra,$t9
lw $v0,32($fp)
beq $v0,$zero,$L90
lw $a0,32($fp)
la $t9,destroy_matrix
jal $ra,$t9
$L90:
lw $v0,24($fp)
beq $v0,$zero,$L92
lw $a0,24($fp)
la $t9,destroy_matrix
jal $ra,$t9
$L92:
lw $v0,28($fp)
beq $v0,$zero,$L93
lw $a0,28($fp)
la $t9,destroy_matrix
jal $ra,$t9
$L93:
lw $v0,cantProcesos
addu $v0,$v0,1
sw $v0,cantProcesos
lw $v0,40($fp)
bne $v0,$zero,$L85
addu $v0,$fp,36
addu $v1,$fp,40
move $a0,$v0
move $a1,$v1
la $t9,leerTamanio
jal $ra,$t9
sw $v0,44($fp)
b $L85
$L86:
sw $zero,48($fp)
```

```
$L77:
lw $v0,48($fp)
move $sp,$fp
lw $ra,64($sp)
lw $fp,60($sp)
addu $sp,$sp,72
j $ra
.end main
.size main, .-main
.ident "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

6. Enunciado

Universidad de Buenos Aires - FIUBA
66.20 Organización de Computadoras
Trabajo práctico 0: Infraestructura básica
1^{er} cuatrimestre de 2016

`$Date: 2016/03/13 20:45:30 $`

1. Objetivos

Familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementando un programa y su correspondiente documentación que resuelvan el problema descripto más abajo.

2. Alcance

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

3. Requisitos

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

4. Recursos

Usaremos el programa GXemul [1] para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión reciente del sistema operativo NetBSD [2].

Durante la primera clase del curso hemos presentado brevemente los pasos necesarios para la instalación y configuración del entorno de desarrollo.

5. Implementación

5.1. Programa

El programa, a escribir en lenguaje C, deberá multiplicar matrices cuadradas de números reales, representados en punto flotante de doble precisión.

Las matrices a multiplicar ingresarán como texto por entrada estándar (`stdin`), donde cáda línea describe completamente cada par de matrices a multiplicar, según el siguiente formato:

$N\ a_{1,1}\ a_{1,2}\ ...\ a_{N,N}\ b_{1,1}\ b_{1,2}\ ...\ b_{N,N}$

La línea anterior representa a las matrices A y B, de NxN. Los elementos de la matriz A son los $a_{x,y}$, siendo x e y los indices de fila y columna respectivamente[1]. Los elementos de la matriz B se representan por los $b_{x,y}$ de la misma forma que los de A.

El fin de línea es el caracter \n (*newline*). Los componentes de la línea están separados entre sí por uno o más espacios. El formato de los números en punto flotante son los que corresponden al especificador de conversión 'g' de `printf`[2].

Por ejemplo, dado el siguiente producto de matrices cuadradas:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

Su representación sería:

```
2 1 2 3 4 5 6 7 8
```

Por cada par de matrices que se presenten por cada línea de entrada, el programa deberá multiplicarlas y presentar el resultado por su salida estándar (`stdout`) en el siguiente formato, hasta que llegue al final del archivo de entrada (`EOF`):

$N\ c_{1,1}\ c_{1,2}\ ...\ c_{N,N}$

Ante un error, el progama deberá informar la situación inmediatamente (por `stderr`) y detener su ejecución.

[1] Notar que es una representación del tipo *row major order*, siguiendo el orden en que C dispone las matrices en memoria.

[2] Ver man 3 printf, "Conversion specifiers".

5.2. Ejemplos

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

```
$ tp0 -h
Usage:
  tp0 -h
  tp0 -V
  tp0 < in_file > out_file
Options:
  -V, --version     Print version and quit.
  -h, --help        Print this information and quit.
Examples:
   tp0 < in.txt > out.txt
   cat in.txt | tp0 > out.txt
```

A continuación, ejecutamos algunas pruebas:

```
$ 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

$ cat example.txt | ./tp0
2 7 10 15 22
3 1 2 3 4 5 6.1 3 2 1
```

En este ejemplo, realizamos las siguientes multiplicaciones, siendo los miembros izquierdos de la ecuación las matrices de entrada (`stdin`), y los miembros derechos las matrices de salida (`stdout`):

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 7 & 10 \\ 15 & 22 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6.1 \\ 3 & 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6.1 \\ 3 & 2 & 1 \end{pmatrix}$$

5.3. Interfaz

Las matrices deberán ser representadas por el tipo de datos `matrix_t`, definido a continuación:

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

Notar que los atributos `rows` y `cols` representan respectivamente la cantidad filas y columnas de la matriz. El atributo `array` contendrá los elementos de la matriz dispuestos en row-major order [3].

Los métodos a implementar, que aplican sobre el tipo de datos `matrix_t` son:

```
// 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);
```

5.4. Portabilidad

Como es usual, es necesario que la implementación desarrollada provea un grado mínimo de portabilidad. Para satisfacer esto, el programa deberá funcionar al menos en NetBSD/pmax (usando el simulador GXemul [1]) y la versión de Linux (Knoppix, RedHat, Debian, Ubuntu) usada para correr el simulador, Linux/i386.

6. Informe

El informe deberá incluir:

- Documentación relevante al diseño e implementación del programa;
- Comando(s) para compilar el programa;
- Las corridas de prueba, con los comentarios pertinentes;
- El código fuente, en lenguaje C;

- El código MIPS32 generado por el compilador[3];
- Este enunciado.

7. Fechas

Fecha de vencimiento: martes 5/4.

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`.

[3] Por motivos prácticos, en la copia impresa sólo es necesario incluir la primera página del código assembly MIPS32 generado por el compilador.