Trabajo Práctico nro. 0: Infraestructura básica

Matias Cerrotta, *Padrón 89992*
matias.cerrotta@gmail.com

2do. Cuatrimestre de 2015
66.20 Organización de Computadoras – Práctica Martes
Facultad de Ingeniería, Universidad de Buenos Aires.

Abstract

Este trabajo práctico nro. 0 busca familiarizar a los alumnos con las herramientas de software que se usarán a lo largo de la cursada. Se implementará un programa en C (y su correspondiente documentación) que resuelverá el problema piloto que se presentará más abajo.

Se mostrará el código C y se obtendrá el código Assembly generado por el compilador *gcc*.

El programa correrá tanto en NetBSD/pmax, usando el simulador GXemul provisto por la cátedra, como en la versión de Linux (Knoppix, RedHat, Debian, Ubuntu) usada para correr el simulador, Linux/i386.

1 Introducción

El programa, a escribir en lenguaje C, deberá multiplicar matrices de números reales, representados en punto flotante de doble precisión. Las matrices a multiplicar ingresarán por entrada estándar (*stdin*), donde cada línea describe una matriz completa en formato de texto.

NxM a1,1 a1,2 ... a1,M a2,1 a2,2 ... a2,M ... aN,1 aN,2 ... aN,M

La línea anterior representa a la matriz A de N filas y M columnas. Los elementos de la matriz A son los a x,y , siendo x e y los indices de fila y columna respectivamente 1 . El fin de línea es el caracter newline. Los componentes de la lńea 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.

Por ejemplo, dada la siguiente matriz:
1 2 3
4 5 6

Su representación sería:
2x3 1 2 3 4 5 6

Por cada par de matrices que se presenten en su entrada, el programa deberá multiplicarlas y presentar el resultado por su salida esándar (stdout) en el mismo formato presentado anteriormente, hasta que llegue al final del archivo de entrada (EOF). Ante un error, el progama deberá informar la situación inmediatamente (por stderr) y detener su ejecución. Tener en cuenta que también se condidera un error que a la entrada se presenten matrices de dimensiones incompatibles entre sí para su multiplicación.

2 Documentación

2.1 Compilación

El programa se compilará con la siguiente instrucción:

```
gcc -Wall -O0 -o tp0 tp0.c
```

Los tests se ejecutarán con el siguiente script:

```
./tests.sh
```

Ejemplo de la salida de ejecución:

```
$ ./tests.sh
Tests #0 success_normal:  OK
Tests #1 sucess_espacios:  OK
Tests #2 error_dimension:  OK
Tests #3 error_max_dimension:  OK
Tests #4 error_max_line:  OK
Tests #5 error_matriz1:  OK
Tests #6 error_matriz2:  OK
```

2.2 Utilización

Opciones de ejecución:

```
-h, --help Print this information.
-V, --version Print version and quit.
```

Ejemplos de ejecución:

```
Examples:
./tp0 -h
./tp0 -V
./tp0 < in_file > out_file
./tp0 < in.txt > out.txt
cat in.txt | tp0 > out.txt
```

A continuación se presenta un ejemplo de prueba:

```
$ cat in.txt
2x3 1 2 3 4 5 6.1
3x2 1 0 0 0 0 1
3x3 1 2 3 4 5 6.1 3 2 1
3x1 1 1 0

$ cat in.txt | ./tp0
2x2 1 3 4 6.1
3x1 3 9 5
```

3 Casos de pruebas

Se crearon los siguientes casos de pruebas:

1. Caso normal.
2. Utilizando espacios entre elementos de matriz.
3. Dimensiones de matrices incompatibles.
4. Superación de caracteres máximos para dimensión.
5. Superación de caracteres máximos para matriz.
6. Elementos de más con respecto a la dimensión.
7. Elementos de menos con respecto a la dimensión.

4 Código fuente

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

#define MAX_LINE_LENGTH 512
#define MAX_DIMENSION_LENGTH 4
#define EXIT_OK 0
#define EXIT_ERROR 1

enum ACCION {
  EMPTY,
  HELP,
  VERSION,
  MULTIPLICAR,
  ERROR
};

void multiplicarMatriz();
enum ACCION procesarArgumentos(int argc, char** argv);

int main(int argc, char **argv) {
  enum ACCION comando = procesarArgumentos(argc, argv);
  switch (comando) {
    case HELP:
      printf("Usage:\n"
      "\ttp0 -h\n"
      "\ttp0 -V\n"
      "\ttp0 < in_file > out_file\n"
      "Options:\n"
      "\t-V, --version\tPrint version and quit.\n"
      "\t-h, --help\tPrint this information.\n"
      "Examples:\n"
      "\ttp0 < in.txt > out.txt\n"
      "\tcat in.txt | tp0 > out.txt\n");
      break;
    case VERSION:
      printf("tp0 v0.1\n");
      break;
    case MULTIPLICAR:
      multiplicarMatriz();
      break;
    case ERROR:
    default:
      return EXIT_ERROR;
  }
  return EXIT_OK;
}

void imprimirMatriz(double** matriz, int filas, int columnas) {
  printf("%dx%d ", filas, columnas);
  int i, j;
  for (i = 0; i < filas; ++i) {
    for (j = 0; j < columnas; ++j) {
      printf("%g ", matriz[i][j]);
    }
  }
  printf("\n");
}

```

```
61 void leerDimension(int* filas, int* columnas) {
62   char c;
63   char* buffer = (char *) malloc(sizeof(char) *
         MAX_DIMENSION_LENGTH);
64   int i = 0, total = 0;
65   while((c = fgetc(stdin)) != EOF) {
66
67     if (c == 'x') {
68       *filas = atoi(buffer);
69       memset(buffer,0,strlen(buffer));
70       i = 0;
71       continue;
72     }
73
74     if(c == ' ') {
75       *columnas = atoi(buffer);
76       break;
77     }
78
79     if(total >= MAX_DIMENSION_LENGTH) {
80       fprintf(stderr, "%s\n", "Tamano de buffer superado para la
             dimension.");
81       free(buffer);
82       exit(EXIT_ERROR);
83     }
84
85     buffer[i] = c;
86     i++;
87     total++;
88   }
89   free(buffer);
90
91   if (c == EOF)
92     exit(EXIT_OK);
93 }
94
95 void leerMatriz(double** matriz, int filas, int columnas) {
96   char c;
97   char* buffer = (char *) malloc(sizeof(char) * MAX_LINE_LENGTH);
98   int i = 0, total = 0, elementos = 0, fila = 0, columna = 0;
99   while((c = fgetc(stdin)) != EOF) {
100
101     total++;
102
103     if (elementos >= filas * columnas) {
104       fprintf(stderr, "%s\n", "Dimension no compatible con datos de
             matriz.");
105       free(buffer);
106       exit(EXIT_ERROR);
107     }
108
109     if (c == ' ' || c == '\n') {
110       // elimino espacios consecutivos
111       if (strlen(buffer) == 0)
112         continue;
113
114       elementos++;
115       matriz[fila][columna] = atof(buffer);
116       memset(buffer,0,strlen(buffer));
117       i = 0;
118       if (columna >= columnas - 1) {
119         fila++;
```

```
120           columna = 0;
121         } else {
122           columna++;
123         }
124
125         if(c == '\n')
126           break;
127
128         continue;
129       }
130
131       if(total >= MAX_LINE_LENGTH) {
132         fprintf(stderr, "%s\n", "Tamano de buffer superado para la
              matriz.");
133         free(buffer);
134         exit(EXIT_ERROR);
135       }
136
137       buffer[i] = c;
138       i++;
139     }
140     free(buffer);
141
142     if (elementos < filas * columnas) {
143       fprintf(stderr, "%s\n", "Dimension no compatible con datos de
              matriz.");
144       exit(EXIT_ERROR);
145     }
146
147
148     if (c == EOF)
149       exit(EXIT_OK);
150 }
151
152 double** crearMatriz(int filas, int columnas) {
153   double** matriz = (double**) malloc(sizeof(double*) * filas);
154   if (matriz == NULL) {
155     fprintf(stderr, "%s\n", "Memoria insuficiente.");
156     exit(EXIT_ERROR);
157   }
158   int i;
159   for(i = 0; i < filas; ++i)
160     matriz[i] = (double*) malloc(sizeof(double) * columnas);
161
162   if (matriz[i - 1] == NULL) {
163     fprintf(stderr, "%s\n", "Memoria insuficiente.");
164     exit(EXIT_ERROR);
165   }
166   return matriz;
167 }
168
169 void multiplicarMatriz() {
170
171   while (1) {
172     int filasA, columnasA;
173     leerDimension(&filasA, &columnasA);
174     double** matrizA = crearMatriz(filasA, columnasA);
175     leerMatriz(matrizA, filasA, columnasA);
176
177     int filasB, columnasB;
178     leerDimension(&filasB, &columnasB);
179     double** matrizB = crearMatriz(filasB, columnasB);
```

```
180     leerMatriz(matrizB, filasB, columnasB);
181
182     if (columnasA != filasB) {
183       fprintf(stderr, "%s\n", "Dimensiones incorrectas de matrices.
              ");
184       free(matrizA);
185       free(matrizB);
186       exit(EXIT_ERROR);
187     }
188
189     double** matrizC = crearMatriz(filasA, columnasB);
190     int i, j, k;
191     for (i = 0; i < filasA; ++i){
192         for (j = 0 ; j < columnasB ; ++j){
193             matrizC[i][j]=0;
194             for ( k = 0; k < columnasA; ++k){
195               matrizC[i][j] = (matrizC[i][j] + (matrizA[i][k] *
                      matrizB[k][j]));
196           }
197       }
198     }
199     imprimirMatriz(matrizC, filasA, columnasB);
200
201     free(matrizA);
202     free(matrizB);
203     free(matrizC);
204   }
205 }
206
207 enum ACCION procesarArgumentos(int argc, char** argv) {
208   // valores por defecto
209   enum ACCION comando = EMPTY;
210
211    /* La funcion getopt obtiene el siguiente argumento especificado
           por argc y argv
212    * mas info: http://www.gnu.org/software/libc/manual/html_node/
           Using-Getopt.html#Using-Getopt
213    * La cadena "hVbri:o:" indica que h, V no tienen argumentos.
214    */
215   int c;
216   while ((c = getopt(argc, argv, "hV")) != -1) {
217     switch (c) {
218       case 'h':
219         comando = HELP;
220         break;
221       case 'V':
222         comando = VERSION;
223         break;
224       default:
225         comando = ERROR;
226         break;
227     }
228   }
229
230   if (comando == EMPTY)
231     comando = MULTIPLICAR;
232
233   return comando;
234 }
```

5 Código $MIPS^{TM}$

```
1   .file 1 "tp0.c"
2   .section .mdebug.abi32
3   .previous
4   .abicalls
5   .rdata
6   .align 2
7 $LC0:
8   .ascii "Usage:\n"
9   .ascii "\ttp0 -h\n"
10  .ascii "\ttp0 -V\n"
11  .ascii "\ttp0 < in_file > out_file\n"
12  .ascii "Options:\n"
13  .ascii "\t-V, --version\tPrint version and quit.\n"
14  .ascii "\t-h, --help\tPrint this information.\n"
15  .ascii "Examples:\n"
16  .ascii "\ttp0 < in.txt > out.txt\n"
17  .ascii "\tcat in.txt | tp0 > out.txt\n\000"
18  .align 2
19 $LC1:
20  .ascii "tp0 v0.1\n\000"
21  .text
22  .align 2
23  .globl main
24  .ent main
25 main:
26  .frame $fp,56,$ra    # vars= 16, regs= 3/0, args= 16, extra= 8
27  .mask 0xd0000000,-8
28  .fmask 0x00000000,0
29  .set noreorder
30  .cpload $t9
31  .set reorder
32  subu $sp,$sp,56
33  .cprestore 16
34  sw $ra,48($sp)
35  sw $fp,44($sp)
36  sw $gp,40($sp)
37  move $fp,$sp
38  sw $a0,56($fp)
39  sw $a1,60($fp)
40  lw $a0,56($fp)
41  lw $a1,60($fp)
42  la $t9,procesarArgumentos
43  jal $ra,$t9
44  sw $v0,24($fp)
45  lw $v0,24($fp)
46  sw $v0,32($fp)
47  li $v0,2     # 0x2
48  lw $v1,32($fp)
49  beq $v1,$v0,$L20
50  lw $v1,32($fp)
51  sltu $v0,$v1,3
52  beq $v0,$zero,$L25
53  li $v0,1     # 0x1
54  lw $v1,32($fp)
55  beq $v1,$v0,$L19
56  b $L23
57 $L25:
58  li $v0,3     # 0x3
59  lw $v1,32($fp)
60  beq $v1,$v0,$L21
```

```
61    b $L23
62  $L19:
63    la  $a0,$LC0
64    la  $t9,printf
65    jal $ra,$t9
66    b $L18
67  $L20:
68    la  $a0,$LC1
69    la  $t9,printf
70    jal $ra,$t9
71    b $L18
72  $L21:
73    la  $t9,multiplicarMatriz
74    jal $ra,$t9
75    b $L18
76  $L23:
77    li  $v0,1       # 0x1
78    sw  $v0,28($fp)
79    b $L17
80  $L18:
81    sw  $zero,28($fp)
82  $L17:
83    lw  $v0,28($fp)
84    move  $sp,$fp
85    lw  $ra,48($sp)
86    lw  $fp,44($sp)
87    addu  $sp,$sp,56
88    j $ra
89    .end  main
90    .size main, .-main
91    .rdata
92    .align  2
93  $LC2:
94    .ascii  "%dx%d \000"
95    .align  2
96  $LC3:
97    .ascii  "%g \000"
98    .align  2
99  $LC4:
100   .ascii  "\n\000"
101   .text
102   .align  2
103   .globl  imprimirMatriz
104   .ent  imprimirMatriz
105 imprimirMatriz:
106   .frame  $fp,48,$ra    # vars= 8, regs= 3/0, args= 16, extra= 8
107   .mask 0xd0000000,-8
108   .fmask  0x00000000,0
109   .set  noreorder
110   .cpload $t9
111   .set  reorder
112   subu  $sp,$sp,48
113   .cprestore 16
114   sw  $ra,40($sp)
115   sw  $fp,36($sp)
116   sw  $gp,32($sp)
117   move  $fp,$sp
118   sw  $a0,48($fp)
119   sw  $a1,52($fp)
120   sw  $a2,56($fp)
121   la  $a0,$LC2
122   lw  $a1,52($fp)
```

```
123     lw   $a2,56($fp)
124     la   $t9,printf
125     jal  $ra,$t9
126     sw   $zero,24($fp)
127 $L27:
128     lw   $v0,24($fp)
129     lw   $v1,52($fp)
130     slt  $v0,$v0,$v1
131     bne  $v0,$zero,$L30
132     b $L28
133 $L30:
134     sw   $zero,28($fp)
135 $L31:
136     lw   $v0,28($fp)
137     lw   $v1,56($fp)
138     slt  $v0,$v0,$v1
139     bne  $v0,$zero,$L34
140     b $L29
141 $L34:
142     lw   $v0,24($fp)
143     sll  $v1,$v0,2
144     lw   $v0,48($fp)
145     addu   $a0,$v1,$v0
146     lw   $v0,28($fp)
147     sll  $v1,$v0,3
148     lw   $v0,0($a0)
149     addu   $v0,$v1,$v0
150     la   $a0,$LC3
151     lw   $a2,0($v0)
152     lw   $a3,4($v0)
153     la   $t9,printf
154     jal  $ra,$t9
155     lw   $v0,28($fp)
156     addu   $v0,$v0,1
157     sw   $v0,28($fp)
158     b $L31
159 $L29:
160     lw   $v0,24($fp)
161     addu   $v0,$v0,1
162     sw   $v0,24($fp)
163     b $L27
164 $L28:
165     la   $a0,$LC4
166     la   $t9,printf
167     jal  $ra,$t9
168     move   $sp,$fp
169     lw   $ra,40($sp)
170     lw   $fp,36($sp)
171     addu   $sp,$sp,48
172     j $ra
173     .end   imprimirMatriz
174     .size imprimirMatriz, .-imprimirMatriz
175     .rdata
176     .align   2
177 $LC5:
178     .ascii   "%s\n\000"
179     .align   2
180 $LC6:
181     .ascii   "Tamano de buffer superado para la dimension.\000"
182     .text
183     .align   2
184     .globl   leerDimension
```

```
185   .ent  leerDimension
186 leerDimension:
187   .frame  $fp,56,$ra     # vars= 16, regs= 3/0, args= 16, extra= 8
188   .mask 0xd0000000,-8
189   .fmask  0x00000000,0
190   .set  noreorder
191   .cpload $t9
192   .set  reorder
193   subu  $sp,$sp,56
194   .cprestore 16
195   sw  $ra,48($sp)
196   sw  $fp,44($sp)
197   sw  $gp,40($sp)
198   move  $fp,$sp
199   sw  $a0,56($fp)
200   sw  $a1,60($fp)
201   li  $a0,4     # 0x4
202   la  $t9,malloc
203   jal $ra,$t9
204   sw  $v0,28($fp)
205   sw  $zero,32($fp)
206   sw  $zero,36($fp)
207 $L36:
208   la  $a0,__sF
209   la  $t9,fgetc
210   jal $ra,$t9
211   sb  $v0,24($fp)
212   lbu $v0,24($fp)
213   sll $v0,$v0,24
214   sra $v1,$v0,24
215   li  $v0,-1      # 0xffffffffffffffff
216   bne $v1,$v0,$L38
217   b $L37
218 $L38:
219   lb  $v1,24($fp)
220   li  $v0,120     # 0x78
221   bne $v1,$v0,$L39
222   lw  $a0,28($fp)
223   la  $t9,atoi
224   jal $ra,$t9
225   lw  $v1,56($fp)
226   sw  $v0,0($v1)
227   lw  $a0,28($fp)
228   la  $t9,strlen
229   jal $ra,$t9
230   lw  $a0,28($fp)
231   move  $a1,$zero
232   move  $a2,$v0
233   la  $t9,memset
234   jal $ra,$t9
235   sw  $zero,32($fp)
236   b $L36
237 $L39:
238   lb  $v1,24($fp)
239   li  $v0,32      # 0x20
240   bne $v1,$v0,$L40
241   lw  $a0,28($fp)
242   la  $t9,atoi
243   jal $ra,$t9
244   move  $v1,$v0
245   lw  $v0,60($fp)
246   sw  $v1,0($v0)
```

```
247   b $L37
248 $L40:
249   lw  $v0,36($fp)
250   slt $v0,$v0,4
251   bne $v0,$zero,$L41
252   la  $a0,__sF+176
253   la  $a1,$LC5
254   la  $a2,$LC6
255   la  $t9,fprintf
256   jal $ra,$t9
257   lw  $a0,28($fp)
258   la  $t9,free
259   jal $ra,$t9
260   li  $a0,1     # 0x1
261   la  $t9,exit
262   jal $ra,$t9
263 $L41:
264   lw  $v1,28($fp)
265   lw  $v0,32($fp)
266   addu  $v1,$v1,$v0
267   lbu $v0,24($fp)
268   sb  $v0,0($v1)
269   lw  $v0,32($fp)
270   addu  $v0,$v0,1
271   sw  $v0,32($fp)
272   lw  $v0,36($fp)
273   addu  $v0,$v0,1
274   sw  $v0,36($fp)
275   b $L36
276 $L37:
277   lw  $a0,28($fp)
278   la  $t9,free
279   jal $ra,$t9
280   lb  $v1,24($fp)
281   li  $v0,-1      # 0xffffffffffffffff
282   bne $v1,$v0,$L35
283   move  $a0,$zero
284   la  $t9,exit
285   jal $ra,$t9
286 $L35:
287   move  $sp,$fp
288   lw  $ra,48($sp)
289   lw  $fp,44($sp)
290   addu  $sp,$sp,56
291   j $ra
292   .end  leerDimension
293   .size leerDimension, .-leerDimension
294   .rdata
295   .align  2
296 $LC7:
297   .ascii  "Dimension no compatible con datos de matriz.\000"
298   .align  2
299 $LC8:
300   .ascii  "Tamano de buffer superado para la matriz.\000"
301   .text
302   .align  2
303   .globl  leerMatriz
304   .ent  leerMatriz
305 leerMatriz:
306   .frame  $fp,72,$ra    # vars= 32, regs= 3/0, args= 16, extra= 8
307   .mask 0xd0000000,-8
308   .fmask  0x00000000,0
```

```
309   .set   noreorder
310   .cpload $t9
311   .set   reorder
312   subu   $sp,$sp,72
313   .cprestore 16
314   sw   $ra,64($sp)
315   sw   $fp,60($sp)
316   sw   $gp,56($sp)
317   move   $fp,$sp
318   sw   $a0,72($fp)
319   sw   $a1,76($fp)
320   sw   $a2,80($fp)
321   li   $a0,512      # 0x200
322   la   $t9,malloc
323   jal  $ra,$t9
324   sw   $v0,28($fp)
325   sw   $zero,32($fp)
326   sw   $zero,36($fp)
327   sw   $zero,40($fp)
328   sw   $zero,44($fp)
329   sw   $zero,48($fp)
330 $L44:
331   la   $a0,__sF
332   la   $t9,fgetc
333   jal  $ra,$t9
334   sb   $v0,24($fp)
335   lbu  $v0,24($fp)
336   sll  $v0,$v0,24
337   sra  $v1,$v0,24
338   li   $v0,-1       # 0xffffffffffffffff
339   bne  $v1,$v0,$L46
340   b  $L45
341 $L46:
342   lw   $v0,36($fp)
343   addu   $v0,$v0,1
344   sw   $v0,36($fp)
345   lw   $v1,76($fp)
346   lw   $v0,80($fp)
347   mult   $v1,$v0
348   mflo   $v1
349   lw   $v0,40($fp)
350   slt  $v0,$v0,$v1
351   bne  $v0,$zero,$L47
352   la   $a0,__sF+176
353   la   $a1,$LC5
354   la   $a2,$LC7
355   la   $t9,fprintf
356   jal  $ra,$t9
357   lw   $a0,28($fp)
358   la   $t9,free
359   jal  $ra,$t9
360   li   $a0,1      # 0x1
361   la   $t9,exit
362   jal  $ra,$t9
363 $L47:
364   lb   $v1,24($fp)
365   li   $v0,32       # 0x20
366   beq  $v1,$v0,$L49
367   lb   $v1,24($fp)
368   li   $v0,10       # 0xa
369   beq  $v1,$v0,$L49
370   b  $L48
```

```
371 $L49:
372   lw   $v0,28($fp)
373   lb   $v0,0($v0)
374   bne $v0,$zero,$L50
375   b $L44
376 $L50:
377   lw   $v0,40($fp)
378   addu  $v0,$v0,1
379   sw   $v0,40($fp)
380   lw   $a0,28($fp)
381   la   $t9,atof
382   jal $ra,$t9
383   lw   $v0,44($fp)
384   sll $v1,$v0,2
385   lw   $v0,72($fp)
386   addu  $a0,$v1,$v0
387   lw   $v0,48($fp)
388   sll $v1,$v0,3
389   lw   $v0,0($a0)
390   addu  $v0,$v1,$v0
391   s.d $f0,0($v0)
392   lw   $a0,28($fp)
393   la   $t9,strlen
394   jal $ra,$t9
395   lw   $a0,28($fp)
396   move  $a1,$zero
397   move  $a2,$v0
398   la   $t9,memset
399   jal $ra,$t9
400   sw   $zero,32($fp)
401   lw   $v0,80($fp)
402   addu  $v1,$v0,-1
403   lw   $v0,48($fp)
404   slt $v0,$v0,$v1
405   bne $v0,$zero,$L51
406   lw   $v0,44($fp)
407   addu  $v0,$v0,1
408   sw   $v0,44($fp)
409   sw   $zero,48($fp)
410   b $L52
411 $L51:
412   lw   $v0,48($fp)
413   addu  $v0,$v0,1
414   sw   $v0,48($fp)
415 $L52:
416   lb   $v1,24($fp)
417   li   $v0,10       # 0xa
418   bne $v1,$v0,$L44
419   b $L45
420 $L48:
421   lw   $v0,36($fp)
422   slt $v0,$v0,512
423   bne $v0,$zero,$L54
424   la   $a0,__sF+176
425   la   $a1,$LC5
426   la   $a2,$LC8
427   la   $t9,fprintf
428   jal $ra,$t9
429   lw   $a0,28($fp)
430   la   $t9,free
431   jal $ra,$t9
432   li   $a0,1      # 0x1
```

```
433   la   $t9,exit
434   jal  $ra,$t9
435 $L54:
436   lw   $v1,28($fp)
437   lw   $v0,32($fp)
438   addu  $v1,$v1,$v0
439   lbu  $v0,24($fp)
440   sb   $v0,0($v1)
441   lw   $v0,32($fp)
442   addu  $v0,$v0,1
443   sw   $v0,32($fp)
444   b $L44
445 $L45:
446   lw   $a0,28($fp)
447   la   $t9,free
448   jal  $ra,$t9
449   lw   $v1,76($fp)
450   lw   $v0,80($fp)
451   mult  $v1,$v0
452   mflo  $v1
453   lw   $v0,40($fp)
454   slt  $v0,$v0,$v1
455   beq  $v0,$zero,$L55
456   la   $a0,__sF+176
457   la   $a1,$LC5
458   la   $a2,$LC7
459   la   $t9,fprintf
460   jal  $ra,$t9
461   li   $a0,1      # 0x1
462   la   $t9,exit
463   jal  $ra,$t9
464 $L55:
465   lb   $v1,24($fp)
466   li   $v0,-1       # 0xffffffffffffffff
467   bne  $v1,$v0,$L43
468   move  $a0,$zero
469   la   $t9,exit
470   jal  $ra,$t9
471 $L43:
472   move  $sp,$fp
473   lw   $ra,64($sp)
474   lw   $fp,60($sp)
475   addu  $sp,$sp,72
476   j $ra
477   .end  leerMatriz
478   .size leerMatriz, .-leerMatriz
479   .rdata
480   .align  2
481 $LC9:
482   .ascii  "Memoria insuficiente.\000"
483   .text
484   .align  2
485   .globl  crearMatriz
486   .ent  crearMatriz
487 crearMatriz:
488   .frame  $fp,48,$ra     # vars= 8, regs= 4/0, args= 16, extra= 8
489   .mask 0xd0010000,-4
490   .fmask  0x00000000,0
491   .set  noreorder
492   .cpload $t9
493   .set  reorder
494   subu  $sp,$sp,48
```

```
495     .cprestore 16
496     sw   $ra,44($sp)
497     sw   $fp,40($sp)
498     sw   $gp,36($sp)
499     sw   $s0,32($sp)
500     move   $fp,$sp
501     sw   $a0,48($fp)
502     sw   $a1,52($fp)
503     lw   $v0,48($fp)
504     sll  $v0,$v0,2
505     move   $a0,$v0
506     la   $t9,malloc
507     jal  $ra,$t9
508     sw   $v0,24($fp)
509     lw   $v0,24($fp)
510     bne  $v0,$zero,$L58
511     la   $a0,__sF+176
512     la   $a1,$LC5
513     la   $a2,$LC9
514     la   $t9,fprintf
515     jal  $ra,$t9
516     li   $a0,1      # 0x1
517     la   $t9,exit
518     jal  $ra,$t9
519 $L58:
520     sw   $zero,28($fp)
521 $L59:
522     lw   $v0,28($fp)
523     lw   $v1,48($fp)
524     slt  $v0,$v0,$v1
525     bne  $v0,$zero,$L62
526     b $L60
527 $L62:
528     lw   $v0,28($fp)
529     sll  $v1,$v0,2
530     lw   $v0,24($fp)
531     addu   $s0,$v1,$v0
532     lw   $v0,52($fp)
533     sll  $v0,$v0,3
534     move   $a0,$v0
535     la   $t9,malloc
536     jal  $ra,$t9
537     sw   $v0,0($s0)
538     lw   $v0,28($fp)
539     addu   $v0,$v0,1
540     sw   $v0,28($fp)
541     b $L59
542 $L60:
543     lw   $v0,28($fp)
544     sll  $v1,$v0,2
545     lw   $v0,24($fp)
546     addu   $v0,$v1,$v0
547     addu   $v0,$v0,-4
548     lw   $v0,0($v0)
549     bne  $v0,$zero,$L63
550     la   $a0,__sF+176
551     la   $a1,$LC5
552     la   $a2,$LC9
553     la   $t9,fprintf
554     jal  $ra,$t9
555     li   $a0,1      # 0x1
556     la   $t9,exit
```

```
557   jal $ra,$t9
558 $L63:
559   lw  $v0,24($fp)
560   move  $sp,$fp
561   lw  $ra,44($sp)
562   lw  $fp,40($sp)
563   lw  $s0,32($sp)
564   addu  $sp,$sp,48
565   j $ra
566   .end  crearMatriz
567   .size crearMatriz, .-crearMatriz
568   .rdata
569   .align  2
570 $LC10:
571   .ascii  "Dimensiones incorrectas de matrices.\000"
572   .text
573   .align  2
574   .globl  multiplicarMatriz
575   .ent  multiplicarMatriz
576 multiplicarMatriz:
577   .frame  $fp,80,$ra    # vars= 40, regs= 3/0, args= 16, extra= 8
578   .mask 0xd0000000,-8
579   .fmask  0x00000000,0
580   .set  noreorder
581   .cpload $t9
582   .set  reorder
583   subu  $sp,$sp,80
584   .cprestore 16
585   sw  $ra,72($sp)
586   sw  $fp,68($sp)
587   sw  $gp,64($sp)
588   move  $fp,$sp
589   .set  noreorder
590   nop
591   .set  reorder
592 $L65:
593   addu  $v0,$fp,28
594   addu  $a0,$fp,24
595   move  $a1,$v0
596   la  $t9,leerDimension
597   jal $ra,$t9
598   lw  $a0,24($fp)
599   lw  $a1,28($fp)
600   la  $t9,crearMatriz
601   jal $ra,$t9
602   sw  $v0,32($fp)
603   lw  $a0,32($fp)
604   lw  $a1,24($fp)
605   lw  $a2,28($fp)
606   la  $t9,leerMatriz
607   jal $ra,$t9
608   addu  $v0,$fp,36
609   addu  $v1,$fp,40
610   move  $a0,$v0
611   move  $a1,$v1
612   la  $t9,leerDimension
613   jal $ra,$t9
614   lw  $a0,36($fp)
615   lw  $a1,40($fp)
616   la  $t9,crearMatriz
617   jal $ra,$t9
618   sw  $v0,44($fp)
```

```
619   lw   $a0,44($fp)
620   lw   $a1,36($fp)
621   lw   $a2,40($fp)
622   la   $t9,leerMatriz
623   jal  $ra,$t9
624   lw   $v1,28($fp)
625   lw   $v0,36($fp)
626   beq  $v1,$v0,$L68
627   la   $a0,__sF+176
628   la   $a1,$LC5
629   la   $a2,$LC10
630   la   $t9,fprintf
631   jal  $ra,$t9
632   lw   $a0,32($fp)
633   la   $t9,free
634   jal  $ra,$t9
635   lw   $a0,44($fp)
636   la   $t9,free
637   jal  $ra,$t9
638   li   $a0,1      # 0x1
639   la   $t9,exit
640   jal  $ra,$t9
641 $L68:
642   lw   $a0,24($fp)
643   lw   $a1,40($fp)
644   la   $t9,crearMatriz
645   jal  $ra,$t9
646   sw   $v0,48($fp)
647   sw   $zero,52($fp)
648 $L69:
649   lw   $v0,52($fp)
650   lw   $v1,24($fp)
651   slt  $v0,$v0,$v1
652   bne  $v0,$zero,$L72
653   b $L70
654 $L72:
655   sw   $zero,56($fp)
656 $L73:
657   lw   $v0,56($fp)
658   lw   $v1,40($fp)
659   slt  $v0,$v0,$v1
660   bne  $v0,$zero,$L76
661   b $L71
662 $L76:
663   lw   $v0,52($fp)
664   sll  $v1,$v0,2
665   lw   $v0,48($fp)
666   addu   $a0,$v1,$v0
667   lw   $v0,56($fp)
668   sll  $v1,$v0,3
669   lw   $v0,0($a0)
670   addu   $v0,$v1,$v0
671   sw   $zero,0($v0)
672   sw   $zero,4($v0)
673   sw   $zero,60($fp)
674 $L77:
675   lw   $v0,60($fp)
676   lw   $v1,28($fp)
677   slt  $v0,$v0,$v1
678   bne  $v0,$zero,$L80
679   b $L75
680 $L80:
```

```
681     lw   $v0,52($fp)
682     sll  $v1,$v0,2
683     lw   $v0,48($fp)
684     addu   $a0,$v1,$v0
685     lw   $v0,56($fp)
686     sll  $v1,$v0,3
687     lw   $v0,0($a0)
688     addu   $a2,$v1,$v0
689     lw   $v0,52($fp)
690     sll  $v1,$v0,2
691     lw   $v0,48($fp)
692     addu   $a0,$v1,$v0
693     lw   $v0,56($fp)
694     sll  $v1,$v0,3
695     lw   $v0,0($a0)
696     addu   $a3,$v1,$v0
697     lw   $v0,52($fp)
698     sll  $v1,$v0,2
699     lw   $v0,32($fp)
700     addu   $a0,$v1,$v0
701     lw   $v0,60($fp)
702     sll  $v1,$v0,3
703     lw   $v0,0($a0)
704     addu   $a1,$v1,$v0
705     lw   $v0,60($fp)
706     sll  $v1,$v0,2
707     lw   $v0,44($fp)
708     addu   $a0,$v1,$v0
709     lw   $v0,56($fp)
710     sll  $v1,$v0,3
711     lw   $v0,0($a0)
712     addu   $v0,$v1,$v0
713     l.d  $f2,0($a1)
714     l.d  $f0,0($v0)
715     mul.d  $f2,$f2,$f0
716     l.d  $f0,0($a3)
717     add.d  $f0,$f0,$f2
718     s.d  $f0,0($a2)
719     lw   $v0,60($fp)
720     addu   $v0,$v0,1
721     sw   $v0,60($fp)
722     b  $L77
723 $L75:
724     lw   $v0,56($fp)
725     addu   $v0,$v0,1
726     sw   $v0,56($fp)
727     b  $L73
728 $L71:
729     lw   $v0,52($fp)
730     addu   $v0,$v0,1
731     sw   $v0,52($fp)
732     b  $L69
733 $L70:
734     lw   $a0,48($fp)
735     lw   $a1,24($fp)
736     lw   $a2,40($fp)
737     la   $t9,imprimirMatriz
738     jal  $ra,$t9
739     lw   $a0,32($fp)
740     la   $t9,free
741     jal  $ra,$t9
742     lw   $a0,44($fp)
```

```
743   la   $t9,free
744   jal  $ra,$t9
745   lw   $a0,48($fp)
746   la   $t9,free
747   jal  $ra,$t9
748   b $L65
749   .end   multiplicarMatriz
750   .size multiplicarMatriz, .-multiplicarMatriz
751   .rdata
752   .align  2
753 $LC11:
754   .ascii  "hV\000"
755   .text
756   .align  2
757   .globl  procesarArgumentos
758   .ent  procesarArgumentos
759 procesarArgumentos:
760   .frame  $fp,56,$ra     # vars= 16, regs= 3/0, args= 16, extra= 8
761   .mask 0xd0000000,-8
762   .fmask  0x00000000,0
763   .set  noreorder
764   .cpload $t9
765   .set  reorder
766   subu  $sp,$sp,56
767   .cprestore 16
768   sw  $ra,48($sp)
769   sw  $fp,44($sp)
770   sw  $gp,40($sp)
771   move  $fp,$sp
772   sw  $a0,56($fp)
773   sw  $a1,60($fp)
774   sw  $zero,24($fp)
775 $L82:
776   lw  $a0,56($fp)
777   lw  $a1,60($fp)
778   la  $a2,$LC11
779   la  $t9,getopt
780   jal $ra,$t9
781   sw  $v0,28($fp)
782   lw  $v1,28($fp)
783   li  $v0,-1       # 0xffffffffffffffff
784   bne $v1,$v0,$L84
785   b $L83
786 $L84:
787   lw  $v0,28($fp)
788   sw  $v0,32($fp)
789   li  $v0,86       # 0x56
790   lw  $v1,32($fp)
791   beq $v1,$v0,$L87
792   li  $v0,104      # 0x68
793   lw  $v1,32($fp)
794   beq $v1,$v0,$L86
795   b $L88
796 $L86:
797   li  $v0,1      # 0x1
798   sw  $v0,24($fp)
799   b $L82
800 $L87:
801   li  $v0,2      # 0x2
802   sw  $v0,24($fp)
803   b $L82
804 $L88:
```

```
805     li   $v0,4      # 0x4
806     sw   $v0,24($fp)
807     b $L82
808 $L83:
809     lw   $v0,24($fp)
810     bne $v0,$zero,$L90
811     li   $v0,3      # 0x3
812     sw   $v0,24($fp)
813 $L90:
814     lw   $v0,24($fp)
815     move   $sp,$fp
816     lw   $ra,48($sp)
817     lw   $fp,44($sp)
818     addu   $sp,$sp,56
819     j $ra
820     .end   procesarArgumentos
821     .size procesarArgumentos, .-procesarArgumentos
822     .ident   "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```