66:20 Organización de Computadoras
Trabajo Práctico 0: Infraestructura Básica

Profesor Titular:
Dr. Ing. José Luis Hamkalo

Docentes:
Ing. Leandro Santi
Ing. Hernán Pérez Masci
Ing. Luciano Natale

Alumnos:
Petalás, Alexis *Padrón Nro. 86742*
Opromolla, Giovanni *Padrón Nro. 87761*
Tapia, Jimena Soledad *Padrón Nro. 88392*

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

Resumen

Con el presente trabajo práctico logramos familiarizarnos con las herramientas a utilizar a lo largo del cuatrimestre para resolver los trabajos propuestos por la cátedra; en particular con el entorno de desarrollo que emula una máquina MIPS corriendo una versión reciente del sistema operativo NetBSD.

Indice

Pág.

1. Introducción **4**

1.1. Objetivo 4

2. Análisis del Problema **5**

2.1. Situación inicial del equipo 5

2.2. Problemática a Resolver 5

3. Diseño e implementación del programa **6**

3.1. Implementación del programa 6

3.2. Esquema de diseño 6

4. Compilación y ejecución **8**

5. Pruebas **10**

5.1. -h Help 10

5.2. -V Version 10

5.3. build/tp0 <data/in1.txt 10

5.4. build/tp0 <data/in2.txt 11

5.5. build/tp0 <data/in3.txt 11

5.6. build/tp0 <data/in4.txt 11

6. Manejo de errores **12**

6.1. Comando inexistente 12

7. Conclusiones **13**

8. Código **14**

8.1. Código C 14

8.2. Código MIPS 20

1. Introducción

Se va a desarrollar un programa en lenguaje C que permita multiplicar matrices de números en punto flotante de doble precisión. Dichas matrices a multiplicar ingresan por entrada estándar y el resultado de la multiplicación tomadas de a pares se muestra por salida estándar (stdout).

1.1. Objetivo

Implementar una función multiplicadora de matrices que sea portable al menos en NetBSD (usando el simulador GXemul [1]) y la versión de Linux (Knoppix, RedHat, Debian, Ubuntu) usada para correr el simulador, Linux/i386.

2. Análisis del Problema

Se detalla a continuación el análisis previo a la resolución del trabajo práctico, clasificado en temas de interés:

2.1. Situación inicial del equipo

1. La tecnología a utilizar es nueva para los integrantes del equipo.
2. Se requiere la configuración y preparación de un entorno de desarrollo específico. También nuevo para los integrantes del equipo.
3. El lenguaje solicitado para programar la solución no es de uso diario de los integrantes del equipo.

De este análisis se resolvió inicialmente focalizarse en el setup del ambiente y luego en el desarrollo propio del trabajo práctico.

2.2. Problemática a Resolver

1. Las matrices de entrada pueden estar mal formadas o incompletas, es decir, pueden contener menos cantidad de valores de los esperados por la dimensión definida.
2. La cantidad de matrices de entrada puede no ser par, interrumpiendo el procesamiento normal de datos ya que el mismo pretende ir multiplicando las matrices de a pares.
3. Las dimensiones indicadas pueden tener un formato incorrecto.
4. Las dimensiones de un par de matrices de entrada, o par formado por la matriz resultante del par anterior, pueden ser incompatibles para su multiplicación.
5. Si el valor en una posición de la matriz es una letra o cualquier caracter no casteable a un número, se considera como un 0.

De este análisis se extrajeron consideraciones para la construcción del parser y el manejo de errores.

3. Diseño e implementación del programa

3.1. Implementación del programa

A partir del problema propuesto, se buscó desarrollar una solución que fuera eficáz y sencilla, centrándose en la reutilización de esta solución como código de entrada para el desarrollo de funciones en MIPS32 en próximos trabajos prácticos.

Se separó la lógica de parseo de comandos de entrada y la de cálculo de multiplicación de las matrices.

3.2. Esquema de diseño

A continuación se muestra un diagrama simplificado de las relaciones de los header de problema propuesto.

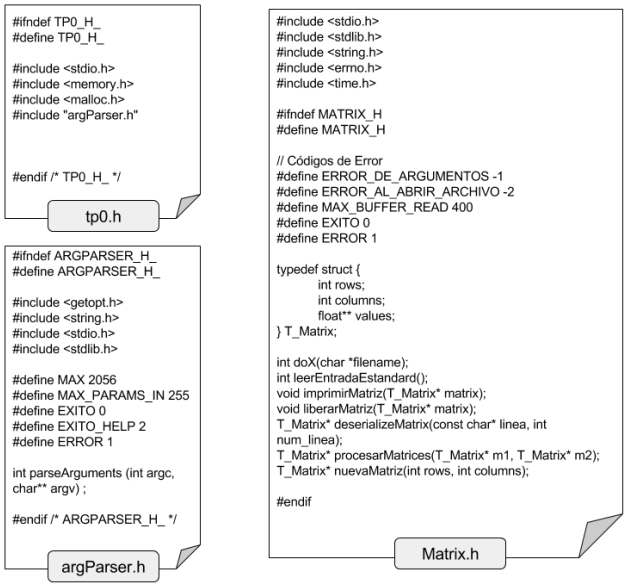

Figura 1: Diagrama interfaces.

Y las relaciones de uso entre las mismas.

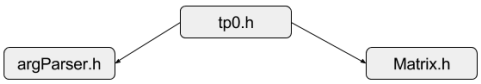

Figura 2: Diagrama Relaciones.

4. Compilación y ejecución

Para la compilación del programa se implementó un Makefile como se puede ver a continuación:

```
1  DEPS = \
2          src/argParser.h \
3          src/Matrix.h \
4          src/tp0.h
5
6  OBJ = build/obj/argParser.o \
7        build/obj/Matrix.o \
8        build/obj/tp0.o
9
10 VIRTUAL = gxemul-6620-20070927
11
12 CC=gcc
13 CP=cp
14 CFLAGS=-I./src -Wall $(ACFLAGS)
15
16 build: prepare tp0
17
18 tp0: $(OBJ)
19   gcc -o build/$@ $(OBJ) $(CFLAGS)
20
21 build/obj/argParser.o: src/argParser.c $(DEPS)
22   $(CC) -c -o $@ src/argParser.c $(CFLAGS)
23
24 build/obj/Matrix.o: src/Matrix.c $(DEPS)
25   $(CC) -c -o $@ src/Matrix.c $(CFLAGS)
26
27 build/obj/tp0.o: src/tp0.c $(DEPS)
28   $(CC) -c -o $@ src/tp0.c $(CFLAGS)
29
30 prepare:
31   -mkdir -p build
32   -mkdir -p build/doc
33   -mkdir -p build/obj
34
35 clean:
36   rm -rf build tags
37
38 virtual-start:
39   sudo ifconfig lo:0 172.20.0.1
40   if [ ! -d ./gxemul/$(VIRTUAL) ]; then bzip2 -dc ./gxemul/$(
         VIRTUAL).tar.bz2 | cpio --sparse -i -v; mv $(VIRTUAL) ./
         gxemul/ ; fi
41   echo "ssh -f -N -R 2222:127.0.0.1:22 $(USER)@172.20.0.1" | xclip
         -sel clip
42   ./gxemul/$(VIRTUAL)/gxemul -e 3max -d ./gxemul/$(VIRTUAL)/netbsd-
         pmax.img
43
44 virtual-reset:
45   rm -rf ./gxemul/$(VIRTUAL)
46
47 virtual-authkey:
48   cat ~/.ssh/id_rsa.pub | ssh -p 2222 root@127.0.0.1 "rm -rf .ssh/
         authorized_keys; mkdir -p ~/.ssh; cat >> ~/.ssh/
         authorized_keys"
49
```

```
50 virtual-deploy:
51   ssh -p 2222 root@127.0.0.1 "rm -rf ~/deploy; mkdir -p ~/deploy;"
52   scp -P 2222 -r makefile src data root@127.0.0.1:/root/deploy
53
54 doc: prepare
55   pandoc README.md -o build/doc/README.pdf
56   pdflatex --output-directory build/doc docs/informe.tex
57   pdflatex --output-directory build/doc docs/informe.tex
58   pdflatex --output-directory build/doc docs/informe.tex
59
60 doc-preview: doc
61   evince build/doc/informe.pdf &
62
63 doc-spell:
64   aspell -t check docs/informe.tex -d es
65
66 export: doc
67   tar -czvf build/entrega_tp0.tar.gz makefile src data -C build/doc
       / informe.pdf README.pdf
```

Para compilar el programa utilizando el Makefile se deben seguir los pasos que se indican en el archivo Readme del proyecto:

```
1  Pasos para correr en la virtual:
2  $ make virtual-start
3  login: root
4  pass: orga6620
5
6  Ctrl+Shift+V o copy-paste de la linea "ssh -f -N -R
       2222:127.0.0.1:22 giovanni@172.20.0.1"
7
8  abrir otra consola y hacer un:
9  $ make virtual-deploy (despliega los archivos importarntes a la
       virtual)
10
11 ahora volvemos a la consolita anterior y entran en deploy
12 $ cd deploy/
13 $ make build
14 Ahi se les compila todo en MIPS...
15
16 el ejecutable se genera en ~/deploy/build/tp0
17 los archivos para correr estan bajo la carpeta de data/
18
19 Ejemplo de corrida: ~/deploy/$ build/tp0 < data/in1.txt
```

5. Pruebas

Para probar la aplicación se ejecutaron las pruebas sencillas presentadas a modo de ejemplo en el enunciado del trabajo.

5.1. -h Help

```
$ build/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
```

5.2. -V Version

```
$ build/tp0 -V

TP0 - Infraestructura Basica
(66.20) Organizacion de las Computadoras
-----------------------------------------
2do Cuatrimestre de 2015
      Version: 1.0

Autores:
        Petalas, Alexis - 86742
        Opromolla, Giovanni - 87761
        Tapia, Jimena Soledad - 88392
```

5.3. build/tp0 <data/in1.txt

Se realiza una prueba simple, con un archivo que contiene un conjunto de matrices cuyas dimensiones son compatibles para la multiplicación de a pares. Se muestra el archivo de entrada y el resultado de la ejecución:

```
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
```

```
$ build/tp0 < data/in1.txt
2x2 1.000000 3.000000 4.000000 6.100000
3x1 3.000000 9.000000 5.000000
```

5.4. build/tp0 <data/in2.txt

Se realiza una prueba con un archivo que contiene un conjunto de matrices cuyas dimensiones no son compatibles para la multiplicación de a pares. Se muestra el archivo de entrada y el resultado de la ejecución:

```
2x2 1 3 4 6.1
3x1 3 9 5
```

```
$ build/tp0 < data/in2.txt
Las propiedades de multiplicacion de matrices no estan satisfechas.
```

5.5. build/tp0 <data/in3.txt

Se realiza una prueba con un archivo que contiene datos con formato incorrecto. No se respeta el formato de entrada acordado "NxM a1,1 a1,2 ... a1,M a2,1 a2,2 ... a2,M ... aN,1 aN,2 ... aN,M". Se muestra el archivo de entrada y el resultado de la ejecución:

```
12xx 12351
```

```
$ build/tp0 < data/in3.txt
Error al leer los valores de NxM en la linea 0.
```

5.6. build/tp0 <data/in4.txt

Se realiza una prueba con un archivo que contiene matrices incompletas, es decir, que las dimensiones indicadas no se corresponden con los valores aX,Y listados. Se muestra el archivo de entrada donde se esperan 6 valores y el resultado de la ejecución:

```
2x2 1.1 2.2 3.3
```

```
$ build/tp0 < data/in4.txt
Faltan valores en la matriz de 2x2 de la linea 0.
```

6. Manejo de errores

Para manejar errores y advertirle al usuario de los mismos se utilizaron mensajes por linea de comando:

6.1. Comando inexistente

En este caso se eligió mostrar el menú de opciones para que el usuario pueda seleccionar un comando correcto, dentro de los entendidos por el programa.

```
$ build/tp0 -u

build/tp0: invalid option -- 'u'
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
```

7. Conclusiones

Luego del desarrollo realizado, logramos familiarizarnos con las herramientas a utilizar a lo largo del cuatrimestre, dejamos el entorno de desarrollo que emula la máquina MIPS funcional y listo para los próximos trabajos prácticos de la materia.

8. Código

8.1. Código C

argParser.h

```
1
2 #ifndef ARGPARSER_H_
3 #define ARGPARSER_H_
4
5 #include <getopt.h>
6 #include <string.h>
7 #include <stdio.h>
8 #include <stdlib.h>
9
10 #define MAX 2056
11 #define MAX_PARAMS_IN 255
12 #define EXITO 0
13 #define EXITO_HELP 2
14 #define ERROR 1
15
16 int parseArguments (int argc, char** argv) ;
17
18 #endif /* ARGPARSER_H_ */
```

argParser.c

```
1
2 #include "argParser.h"
3
4 int showMenuHelp()
5 {
6     printf("Usage:\n");
7     printf("  ./tp0 -h\n");
8     printf("  ./tp0 -V\n");
9     printf("  ./tp0  < in_file > out_file\n");
10    printf("Options:\n");
11    printf("  -V, --version\tPrint version and quit.\n");
12    printf("  -h, --help\tPrint this information and quit.\n");
13    printf("Examples:\n");
14    printf("  ./tp0 < in.txt > out.txt\n");
15    printf("  cat in.txt | ./tp0 > out.txt\n");
16    return EXITO;
17 }
18
19 int showMenuVersion()
20 {
21   printf("\n\tTP0 - Infraestructura Basica\n");
22   printf("  (66.20) Organizacion de las Computadoras\n");
23   printf("-----------------------------------------------------\n");
24   printf("\t2do Cuatrimestre de 2015\n");
25   printf("\t      Version: 1.0\n");
26   printf("\nAutores:\n");
27   printf("          Petalas, Alexis - 86742\n");
28   printf("          Opromolla, Giovanni - 87761\n");
29   printf("          Tapia, Jimena Soledad - 88392\n\n");
30   return EXITO;
31
32 }
```

```
33
34
35 int readOptions (int argc, char** argv, const char* op_cortas,
       const struct option op_largas[]) {
36
37   int siguiente_opcion = 0;
38   int result = ERROR;
39
40   siguiente_opcion = getopt_long(argc, argv, op_cortas, op_largas,
         NULL);
41
42   switch (siguiente_opcion) {
43     case 'h' : /* -h o --help */
44       showMenuHelp();
45       result = EXITO_HELP;
46       break;
47     case 'V' : /* -V o --version */
48       showMenuVersion();
49       result = EXITO_HELP;
50       break;
51     case '?' : /* opcion no valida */
52       showMenuHelp();
53       result = EXITO_HELP;
54       break;
55     default : /* Algo mas? No esperado. Abortamos */
56       result = EXITO;
57   }
58
59   return result;
60 }
61
62
63
64 int parseArguments (int argc, char** argv) {
65
66   int result = ERROR;
67
68   /* Una cadena que lista las opciones cortas validas */
69   const char* op_cortas = "hV" ;
70
71   /* Una estructura de varios arrays describiendo los valores
         largos */
72   const struct option op_largas[] = {
73     { "help",          0,  NULL,   'h'},
74     { "version",       0,  NULL,   'V'},
75     { NULL,            0,  NULL,   0  }
76   };
77
78   result = readOptions(argc, argv, op_cortas, op_largas);
79
80   if(result == ERROR){
81       showMenuHelp();
82   }
83
84   return result;
85 }
```

Matrix.h

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
```

```
4  #include <errno.h>
5  #include <time.h>
6
7  #ifndef MATRIX_H
8  #define MATRIX_H
9
10 // Codigos de Error
11 #define ERROR_DE_ARGUMENTOS -1
12 #define ERROR_AL_ABRIR_ARCHIVO -2
13 #define MAX_BUFFER_READ 400
14 #define EXITO 0
15 #define ERROR 1
16
17 typedef struct {
18   int rows;
19   int columns;
20   float** values;
21 } T_Matrix;
22
23 int doX(char *filename);
24 int leerEntradaEstandard();
25 void imprimirMatriz(T_Matrix* matrix);
26 void liberarMatriz(T_Matrix* matrix);
27 T_Matrix* deserializeMatrix(const char* linea, int num_linea);
28 T_Matrix* procesarMatrices(T_Matrix* m1, T_Matrix* m2);
29 T_Matrix* nuevaMatriz(int rows, int columns);
30
31 #endif
```

Matrix.c

```
1
2  #include "Matrix.h"
3
4  #define BUF_SIZE 255
5
6  int leerEntradaEstandard()
7  {
8
9      char line[BUF_SIZE];
10     int num_linea = 0;
11
12     T_Matrix* m1 = NULL;
13     T_Matrix* m2 = NULL;
14     T_Matrix* m_resultado = NULL;
15     /** Lectura de stdin para obtener las matrices linea por linea
           **/
16
17     while((fgets(line,BUF_SIZE,stdin)) != NULL){
18
19       if(strlen(line) > 2){
20       if (num_linea%2 == 0)    /** Pares **/
21       {
22         m1 = deserializeMatrix(line, num_linea);
23         if (m1 == NULL) {
24           /** Algun problema surgio al deserealizar la matriz **/
25           fprintf(stderr, "Ocurrio un error al procesar una las
                 lineas. Verique el formato y la cantidad de matrices
                 en el archivo.\n");
26           break;
27         }
28       }
```

```
29        else              /** Impares **/
30        {
31          m2 = deserializeMatrix(line, num_linea);
32          if (m2 == NULL){
33            liberarMatriz(m1);
34            fprintf(stderr, "Ocurrio un error al procesar una las
                  lineas. Verique el formato y la cantidad de matrices
                  en el archivo.\n");
35            break;
36          }   /** Algun problema surgio al deserealizar la matriz **/
37          else{
38            m_resultado = procesarMatrices(m1, m2);
39
40            if(m_resultado != NULL)
41              imprimirMatriz(m_resultado);
42
43            liberarMatriz(m1);
44            liberarMatriz(m2);
45            liberarMatriz(m_resultado);
46          }
47        }
48        ++num_linea;
49        }
50    }
51
52      return EXITO;
53  }
54
55  void imprimirMatriz(T_Matrix* matrix){
56    int row, columns;
57    printf("%dx%d ", matrix->rows, matrix->columns);
58    for (row=0; row<matrix->rows; row++)
59    {
60        for(columns=0; columns<matrix->columns; columns++)
61             printf("%f ", matrix->values[row][columns]);
62    }
63    printf("\n");
64  }
65
66  void liberarMatriz(T_Matrix* matrix){
67
68    int row;
69
70    for( row=0; row<matrix->rows; row++ ) {
71        free( matrix->values[row] );
72    }
73    free( matrix->values );
74
75    free(matrix);
76    matrix = NULL;
77  }
78
79
80  char* serializeMatrix(T_Matrix* m ){
81      char* matrix_str = NULL;
82
83      return matrix_str;
84  }
85
86  T_Matrix* deserializeMatrix(const char* linea, int num_linea){
87
88    int rows = -1;
```

```
89    int columns = -1;
90    int val = sscanf(linea, "%dx%d", &rows, &columns);
91
92    /** Verificar que se hayan podido leer ambos valores **/
93    if (val != 2)
94    {
95      fprintf(stderr, "Error al leer los valores de NxM en la linea %
          d.\n", num_linea);
96      exit(1);
97    }
98    /** Verificar que ambos valores sean mayores a cero **/
99    if((rows < 0) || (columns < 0)){
100     fprintf(stderr, "Los valres de NxM deben ser valores positivos.
            Linea conflictiva: %d.\n", num_linea);
101     exit(1);
102   }
103
104   /** Creo matriz y aloco su memoria **/
105   T_Matrix* matrix = nuevaMatriz(rows, columns);
106
107   /** Obtengo los valores de la matriz **/
108     char* valores = strstr (linea," ");;
109     char *token;
110     token = strtok(valores, " ");
111
112     int f=0, c=0;
113     while( (token != NULL) && (f < rows ) )
114     {
115       matrix->values[f][c] = atof(token);
116       token = strtok(NULL, " ");
117       if(f < rows){
118         c++;
119         if(c == columns){
120           f++;
121           c = 0;
122         }
123       }
124     }
125
126     /** Matriz incompleta **/
127     if(f != rows){
128       fprintf(stderr, "Faltan valores en la matriz de %dx%d de la
              linea %d.\n", matrix->rows, matrix->columns, num_linea );
129       liberarMatriz(matrix);
130       exit(1);
131     }
132
133     /** Imprimo la matriz **/
134     //imprimirMatriz(matrix);
135
136   return matrix;
137 }
138
139 T_Matrix* nuevaMatriz(int rows, int columns){
140
141   T_Matrix* matrix = (T_Matrix*) malloc(sizeof (T_Matrix) );
142
143   matrix->rows = rows;
144   matrix->columns = columns;
145
146   /** Aloco espacio para la matriz **/
147   matrix->values = (float**) malloc (sizeof(float*)*matrix->rows);
```

```
148   int i = 0;
149   for(; i < matrix->rows; ++i){
150     matrix->values[i] = (float*) malloc (sizeof(float)*matrix->
          columns);
151   }
152
153   return matrix;
154 }
155
156 T_Matrix* procesarMatrices(T_Matrix* m1, T_Matrix* m2){
157
158   T_Matrix* matrix = NULL;
159
160   if(m1->columns == m2->rows){
161
162     /** Creo matriz y aloco su memoria **/
163     matrix = nuevaMatriz(m1->rows, m2->columns);
164
165     int row1, column2,  k;
166     float sum;
167
168       for(row1=0; row1<m1->rows; ++row1) //filas de la primer
            matriz
169       {
170         for(column2=0; column2<m2->columns; ++column2)  //columnas
              de la segunda matriz
171         {
172           sum=0;
173
174           for(k=0;k<m1->columns;k++)
175             sum=sum + m1->values[row1][k] * m2->values[k][column2];
176
177           matrix->values[row1][column2]=sum;
178         }
179       }
180
181   }else{
182       fprintf(stderr, "Las propiedades de multiplicacion de
            matrices no estan satisfechas.\n");
183       exit(1);
184   }
185
186   return matrix;
187
188 }
```

tp0.h

```
1  /*
2   * tp0.h
3   *
4   *  Created on: Sep 9, 2015
5   *      Author: giovanni
6   */
7
8  #ifndef TP0_H_
9  #define TP0_H_
10
11 #include <stdio.h>
12 #include <memory.h>
13 #include <malloc.h>
14 #include "argParser.h"
```

```
15
16
17
18 #endif /* TP0_H_ */
```

tp0.c

```
1 #include "tp0.h"
2 #include "Matrix.h"
3
4 int main(int argc, char** argv){
5
6       int result = parseArguments(argc, argv);
7
8       if(result == 0)
9     result = leerEntradaEstandard();
10      else if (result == 1)
11    perror("Error al obtener los argumentos.");
12
13      return 0;
14 }
```

8.2. Código MIPS

Porción de código ilustrativa - argParser.o

```
1
2 argParser.o:     file format elf32-tradlittlemips
3
4 Disassembly of section .text:
5
6 00000000 <showMenuHelp>:
7    0: 3c1c0000  lui gp,0x0
8    4: 279c0000  addiu gp,gp,0
9    8: 0399e021  addu  gp,gp,t9
10   c: 27bdffd8  addiu sp,sp,-40
11  10: afbc0010  sw  gp,16(sp)
12  14: afbf0020  sw  ra,32(sp)
13  18: afbe001c  sw  s8,28(sp)
14  1c: afbc0018  sw  gp,24(sp)
15  20: 03a0f021  move  s8,sp
16  24: 8f840000  lw  a0,0(gp)
17  28: 00000000  nop
18  2c: 24840000  addiu a0,a0,0
19  30: 8f990000  lw  t9,0(gp)
20  34: 00000000  nop
21  38: 0320f809  jalr  t9
22  3c: 00000000  nop
23  40: 8fdc0010  lw  gp,16(s8)
24  44: 00000000  nop
25  48: 8f840000  lw  a0,0(gp)
26  4c: 00000000  nop
27  50: 24840008  addiu a0,a0,8
28  54: 8f990000  lw  t9,0(gp)
29  58: 00000000  nop
30  5c: 0320f809  jalr  t9
31  60: 00000000  nop
```

```
32   64: 8fdc0010   lw   gp,16(s8)
33   68: 00000000   nop
34   6c: 8f840000   lw   a0,0(gp)
35   70: 00000000   nop
36   74: 24840014   addiu a0,a0,20
37   78: 8f990000   lw   t9,0(gp)
38   7c: 00000000   nop
39   80: 0320f809   jalr  t9
40   84: 00000000   nop
41   88: 8fdc0010   lw   gp,16(s8)
```

Porción de código ilustrativa - Matrix.o

```
1
2  Matrix.o:     file format elf32-tradlittlemips
3
4  Disassembly of section .text:
5
6  00000000 <leerEntradaEstandard>:
7     0: 3c1c0000   lui gp,0x0
8     4: 279c0000   addiu gp,gp,0
9     8: 0399e021   addu  gp,gp,t9
10    c: 27bdfec8   addiu sp,sp,-312
11   10: afbc0010   sw   gp,16(sp)
12   14: afbf0130   sw   ra,304(sp)
13   18: afbe012c   sw   s8,300(sp)
14   1c: afbc0128   sw   gp,296(sp)
15   20: 03a0f021   move  s8,sp
16   24: afc00118   sw   zero,280(s8)
17   28: afc0011c   sw   zero,284(s8)
18   2c: afc00120   sw   zero,288(s8)
19   30: afc00124   sw   zero,292(s8)
20   34: 27c40018   addiu a0,s8,24
21   38: 240500ff   li   a1,255
22   3c: 8f860000   lw   a2,0(gp)
23   40: 8f990000   lw   t9,0(gp)
24   44: 00000000   nop
25   48: 0320f809   jalr  t9
26   4c: 00000000   nop
27   50: 8fdc0010   lw   gp,16(s8)
28   54: 14400003   bnez  v0,64 <leerEntradaEstandard+0x64>
29   58: 00000000   nop
30   5c: 10000071   b 224 <leerEntradaEstandard+0x224>
31   60: 00000000   nop
32   64: 27c40018   addiu a0,s8,24
33   68: 8f990000   lw   t9,0(gp)
34   6c: 00000000   nop
35   70: 0320f809   jalr  t9
36   74: 00000000   nop
37   78: 8fdc0010   lw   gp,16(s8)
38   7c: 2c420003   sltiu v0,v0,3
39   80: 1440ffec   bnez  v0,34 <leerEntradaEstandard+0x34>
40   84: 00000000   nop
41   88: 8fc20118   lw   v0,280(s8)
```

Porción de código ilustrativa - tp0.o

```
1
2  tp0.o:     file format elf32-tradlittlemips
3
```

```
Disassembly of section .text:

00000000 <main>:
   0: 3c1c0000  lui gp,0x0
   4: 279c0000  addiu gp,gp,0
   8: 0399e021  addu  gp,gp,t9
   c: 27bdffd0  addiu sp,sp,-48
  10: afbc0010  sw  gp,16(sp)
  14: afbf0028  sw  ra,40(sp)
  18: afbe0024  sw  s8,36(sp)
  1c: afbc0020  sw  gp,32(sp)
  20: 03a0f021  move  s8,sp
  24: afc40030  sw  a0,48(s8)
  28: afc50034  sw  a1,52(s8)
  2c: 8fc40030  lw  a0,48(s8)
  30: 8fc50034  lw  a1,52(s8)
  34: 8f990000  lw  t9,0(gp)
  38: 00000000  nop
  3c: 0320f809  jalr  t9
  40: 00000000  nop
  44: 8fdc0010  lw  gp,16(s8)
  48: afc20018  sw  v0,24(s8)
  4c: 8fc20018  lw  v0,24(s8)
  50: 00000000  nop
  54: 14400008  bnez  v0,78 <main+0x78>
  58: 00000000  nop
  5c: 8f990000  lw  t9,0(gp)
  60: 00000000  nop
  64: 0320f809  jalr  t9
  68: 00000000  nop
  6c: 8fdc0010  lw  gp,16(s8)
  70: 1000000d  b a8 <main+0xa8>
  74: afc20018  sw  v0,24(s8)
  78: 8fc30018  lw  v1,24(s8)
  7c: 24020001  li  v0,1
  80: 14620009  bne v1,v0,a8 <main+0xa8>
  84: 00000000  nop
  88: 8f840000  lw  a0,0(gp)
```

Referencias

[1] GXemul,http://gavare.se/gxemul/.

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

[3] time man page http://unixhelp.ed.ac.uk/CGI/man-cgi?time.

[4] GNU gprof http://www.cs.utah.edu/dept/old/texinfo/as/gprof.html.