66:20 Organización de Computadoras
Trabajo Práctico 1: assembly MIPS

Profesor Titular:
Dr. Ing. José Luis Hamkalo

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

Alumnos:
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 el conjunto de instrucciones MIPS y el concepto de ABI que se utiliza en la cátedra.

Indice

Pág.

1. Introducción

Se va a implementar una función en MIPS que resuelva la multiplicación de matrices de números en punto flotante de doble precisión. Dicha función fue previamente desarrollada en lenguaje C como parte en un programa que recibe matrices por entrada estándar, las multiplica tomadas de a pares y devuelve el resultado de la multiplicación por salida estándar (stdout).

1.1. Objetivo

Transcribir e implementar a lenguaje MIPS una función multiplicadora de matrices previamente desarrollada en lenguaje C que además debe resolver la funcionalidad original.

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 comprensión y manipulación de la ABI y el conjunto de instrucciones propuesto por la cátedra.
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 la comprensión de la ABI y la práctica del lenguaje MIPS.

2.2. Problemática a Resolver

Debe mantenerse del desarrollo original aprobado en el trabajo práctico anterior:

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.

Se agrega en este trabajo práctico:

1. El parseo de los archivos de entrada se va a dejar dentro de la implementación en lenguaje C.
2. El manejo de memoria de las matrices se va a dejar dentro de la implementación en lenguaje C.

3. El manejo de errores se va a dejar dentro de la implementación en lenguaje C.

De este análisis se extrajeron nuevas consideraciones para la reutilización del desarrollo entregado anteriormente y la implementación de la función en MIPS.

3. Diseño e implementación del programa

3.1. Implementación del programa

Para construir este trabajo práctico, se mantuvo la separación de la lógica de parseo de comandos de entrada y la de cálculo de multiplicación de las matrices, propuesta en la solución de base.

Si bien dicha solución propuesta para el trabajo práctico anterior fue eficáz y sencilla, se decidió simplificar aún más la función que efectúa la multiplicación para lograr exponer una interfáz simple entre el módulo desarrollado en C y el módulo desarrollado en MIPS.

3.2. Esquema de diseño

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

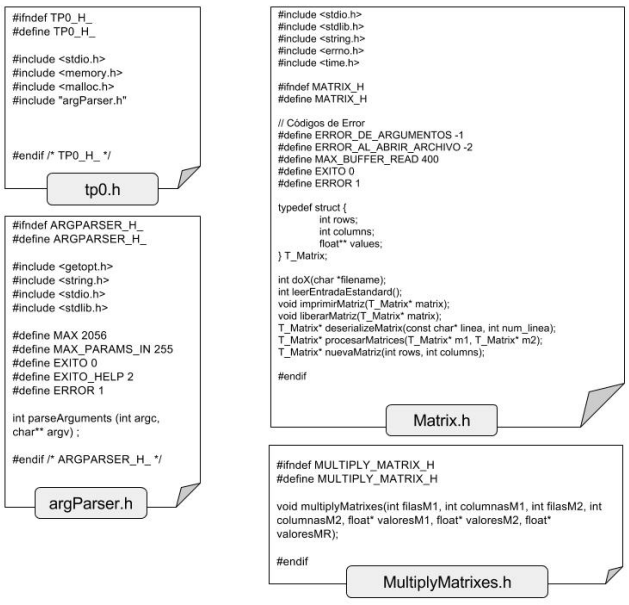

Figura 1: Diagrama interfaces.

Y las relaciones de uso entre las mismas.

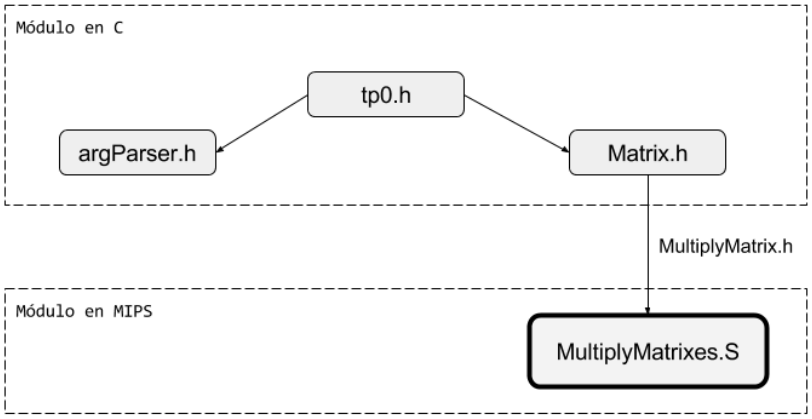

Figura 2: Diagrama Relaciones.

3.3. Función MIPS

Para la implementación del módulo de MIPS se separó primero la lógica relacionada con la multiplicación de matrices en la función que se detalla a continuación, para luego transcribirla a lenguaje MIPS:

```
1  void multiplyMatrixes(int filasM1, int columnasM1, int filasM2, int
       columnasM2, float* valoresM1, float* valoresM2, float*
     valoresMR) {
2
3    int row1, column2,  k;
4    float sum;
5
6    for(row1=0; row1<filasM1; ++row1) //filas de la primer matriz
7    {
8        for(column2=0; column2<columnasM2; ++column2)  //columnas de
             la segunda matriz
9        {
10         sum=0;
11
12         for(k=0;k<columnasM1;k++)
13           sum=sum + valoresM1[(row1*columnasM1) + k] * valoresM2[(k
                 *columnasM2) + column2];
14
15         valoresMR[(row1*columnasM2) + column2]=sum;
16       }
17   }
18 }
```

El stack para esta función implementada en MIPS queda como se grafica a continuación:

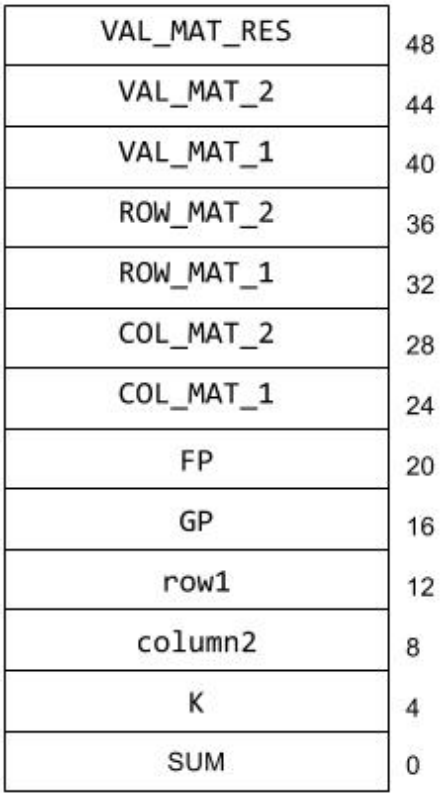

Figura 3: Diagrama de stack.

4. Compilación y ejecución

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

```
DEPS = \
        src/argParser.h \
        src/MultiplyMatrix.h \
        src/Matrix.h \
        src/tp0.h

OBJ = build/obj/argParser.o \
     build/obj/MultiplyMatrix.o \
       build/obj/Matrix.o \
       build/obj/tp0.o

VIRTUAL = gxemul-6620-20070927

CC=gcc
CP=cp
CFLAGS=-I./src -Wall $(ACFLAGS)

build: prepare tp0

tp0: $(OBJ)
  gcc -o build/$@ $(OBJ) $(CFLAGS)

build/obj/argParser.o: src/argParser.c $(DEPS)
  $(CC) -c -o $@ src/argParser.c $(CFLAGS)

build/obj/MultiplyMatrix.o: src/MultiplyMatrix.S $(DEPS)
  $(CC) -c -o $@ src/MultiplyMatrix.S $(CFLAGS)

build/obj/Matrix.o: src/Matrix.c $(DEPS)
  $(CC) -c -o $@ src/Matrix.c $(CFLAGS)

build/obj/tp0.o: src/tp0.c $(DEPS)
  $(CC) -c -o $@ src/tp0.c $(CFLAGS)

prepare:
  -mkdir -p build
  -mkdir -p build/doc
  -mkdir -p build/obj

clean:
  rm -rf build tags

virtual-start:
  sudo ifconfig lo:0 172.20.0.1
  if [ ! -d ./gxemul/$(VIRTUAL) ]; then bzip2 -dc ./gxemul/$(
      VIRTUAL).tar.bz2 | cpio --sparse -i -v; mv $(VIRTUAL) ./
      gxemul/ ; fi
  echo "ssh -f -N -R 2222:127.0.0.1:22 $(USER)@172.20.0.1" | xclip
      -sel clip
  ./gxemul/$(VIRTUAL)/gxemul -e 3max -d ./gxemul/$(VIRTUAL)/netbsd-
      pmax.img

virtual-reset:
  rm -rf ./gxemul/$(VIRTUAL)

```

```
52 virtual-authkey:
53   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"
54
55 virtual-deploy:
56   ssh -p 2222 root@127.0.0.1 "rm -rf ~/deploy; mkdir -p ~/deploy;"
57   scp -P 2222 -r makefile src data root@127.0.0.1:/root/deploy
58
59 doc: prepare
60   pandoc README.md -o build/doc/README.pdf
61   pdflatex --output-directory build/doc docs/informe.tex
62   pdflatex --output-directory build/doc docs/informe.tex
63   pdflatex --output-directory build/doc docs/informe.tex
64
65 doc-preview: doc
66   evince build/doc/informe.pdf &
67
68 doc-spell:
69   aspell -t check docs/informe.tex -d es
70
71 export: doc
72   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/tp1
17 los archivos para correr estan bajo la carpeta de data/
18
19 Ejemplo de corrida: ~/deploy/$ build/tp1 < 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/tp1 -h
Usage:
  ./tp1 -h
  ./tp1 -V
  ./tp1 < in_file > out_file
Options:
  -V, --version Print version and quit.
  -h, --help    Print this information and quit.
Examples:
  ./tp1 < in.txt > out.txt
  cat in.txt | ./tp1 > out.txt
```

5.2. -V Version

```
$ ./build/tp1 -V

        TP1 - assembly MIPS
  (66.20) Organizacion de las Computadoras
-----------------------------------------------
        2do Cuatrimestre de 2015
              Version: 1.0

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

5.3. build/tp1 <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/tp1 < data/in1.txt
2x2 1.000000 3.000000 4.000000 6.100000
3x1 3.000000 9.000000 5.000000
```

5.4. build/tp1 <data/in5.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:

```
1 4x3 1 2 3 4 5 6 7 8 9 10 11 12
2 3x1 1 2 3
```

```
1 $ ./build/tp1 < data/in5.txt
2 4x1 14.000000 32.000000 50.000000 68.000000
```

5.5. build/tp1 <data/in8.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:

```
1 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
      25
2 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
```

```
1 ./build/tp1 < data/in8.txt
2 5x5 175.000000 160.000000 145.000000 130.000000 115.499992
      550.000000 510.000000 470.000000 430.000000 390.999969
      925.000000 860.000000 795.000000 730.000000 666.499939
      1300.000000 1210.000000 1120.000000 1030.000000 941.999939
      1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
```

5.6. build/tp1 <data/in9.txt

Se realiza una prueba simple, con un archivo que contiene un varios 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:

```
1 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
      25
2 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
3 2x3 1 2 3 4 5 6.1
4 3x2 1 0 0 0 0 1
5 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
      25
6 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
7 2x3 1 2 3 4 5 6.1
8 3x2 1 0 0 0 0 1
9 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
      25
```

```
10 | 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
11 | 2x3 1 2 3 4 5 6.1
12 | 3x2 1 0 0 0 0 1
13 | 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
     25
14 | 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
15 | 2x3 1 2 3 4 5 6.1
16 | 3x2 1 0 0 0 0 1
17 | 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
     25
18 | 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
19 | 2x3 1 2 3 4 5 6.1
20 | 3x2 1 0 0 0 0 1
21 | 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
     25
22 | 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
23 | 2x3 1 2 3 4 5 6.1
24 | 3x2 1 0 0 0 0 1
25 | 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
     25
26 | 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
27 | 2x3 1 2 3 4 5 6.1
28 | 3x2 1 0 0 0 0 1
29 | 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
     25
30 | 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
31 | 2x3 1 2 3 4 5 6.1
32 | 3x2 1 0 0 0 0 1
33 | 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
     25
34 | 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
35 | 2x3 1 2 3 4 5 6.1
36 | 3x2 1 0 0 0 0 1
37 | 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
     25
38 | 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
39 | 2x3 1 2 3 4 5 6.1
40 | 3x2 1 0 0 0 0 1
41 | 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
     25
42 | 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
43 | 2x3 1 2 3 4 5 6.1
44 | 3x2 1 0 0 0 0 1
45 | 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
     25
46 | 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
47 | 2x3 1 2 3 4 5 6.1
48 | 3x2 1 0 0 0 0 1
49 | 5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
     25
50 | 5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
       1.1
```

```
2x3 1 2 3 4 5 6.1
3x2 1 0 0 0 0 1
5x5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
    25
5x5 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
    1.1
```

```
$ ./build/tp1 < data/in9.txt
5x5 175.000000 160.000000 145.000000 130.000000 115.499992
    550.000000 510.000000 470.000000 430.000000 390.999969
    925.000000 860.000000 795.000000 730.000000 666.499939
    1300.000000 1210.000000 1120.000000 1030.000000 941.999939
    1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
2x2 1.000000 3.000000 4.000000 6.100000
5x5 175.000000 160.000000 145.000000 130.000000 115.499992
    550.000000 510.000000 470.000000 430.000000 390.999969
    925.000000 860.000000 795.000000 730.000000 666.499939
    1300.000000 1210.000000 1120.000000 1030.000000 941.999939
    1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
2x2 1.000000 3.000000 4.000000 6.100000
5x5 175.000000 160.000000 145.000000 130.000000 115.499992
    550.000000 510.000000 470.000000 430.000000 390.999969
    925.000000 860.000000 795.000000 730.000000 666.499939
    1300.000000 1210.000000 1120.000000 1030.000000 941.999939
    1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
2x2 1.000000 3.000000 4.000000 6.100000
5x5 175.000000 160.000000 145.000000 130.000000 115.499992
    550.000000 510.000000 470.000000 430.000000 390.999969
    925.000000 860.000000 795.000000 730.000000 666.499939
    1300.000000 1210.000000 1120.000000 1030.000000 941.999939
    1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
2x2 1.000000 3.000000 4.000000 6.100000
5x5 175.000000 160.000000 145.000000 130.000000 115.499992
    550.000000 510.000000 470.000000 430.000000 390.999969
    925.000000 860.000000 795.000000 730.000000 666.499939
    1300.000000 1210.000000 1120.000000 1030.000000 941.999939
    1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
2x2 1.000000 3.000000 4.000000 6.100000
5x5 175.000000 160.000000 145.000000 130.000000 115.499992
    550.000000 510.000000 470.000000 430.000000 390.999969
    925.000000 860.000000 795.000000 730.000000 666.499939
    1300.000000 1210.000000 1120.000000 1030.000000 941.999939
    1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
2x2 1.000000 3.000000 4.000000 6.100000
5x5 175.000000 160.000000 145.000000 130.000000 115.499992
    550.000000 510.000000 470.000000 430.000000 390.999969
    925.000000 860.000000 795.000000 730.000000 666.499939
    1300.000000 1210.000000 1120.000000 1030.000000 941.999939
    1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
2x2 1.000000 3.000000 4.000000 6.100000
5x5 175.000000 160.000000 145.000000 130.000000 115.499992
    550.000000 510.000000 470.000000 430.000000 390.999969
    925.000000 860.000000 795.000000 730.000000 666.499939
    1300.000000 1210.000000 1120.000000 1030.000000 941.999939
    1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
2x2 1.000000 3.000000 4.000000 6.100000
5x5 175.000000 160.000000 145.000000 130.000000 115.499992
    550.000000 510.000000 470.000000 430.000000 390.999969
    925.000000 860.000000 795.000000 730.000000 666.499939
    1300.000000 1210.000000 1120.000000 1030.000000 941.999939
    1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
2x2 1.000000 3.000000 4.000000 6.100000
```

```
20 5x5 175.000000 160.000000 145.000000 130.000000 115.499992
       550.000000 510.000000 470.000000 430.000000 390.999969
       925.000000 860.000000 795.000000 730.000000 666.499939
       1300.000000 1210.000000 1120.000000 1030.000000 941.999939
       1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
21 2x2 1.000000 3.000000 4.000000 6.100000
22 5x5 175.000000 160.000000 145.000000 130.000000 115.499992
       550.000000 510.000000 470.000000 430.000000 390.999969
       925.000000 860.000000 795.000000 730.000000 666.499939
       1300.000000 1210.000000 1120.000000 1030.000000 941.999939
       1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
23 2x2 1.000000 3.000000 4.000000 6.100000
24 5x5 175.000000 160.000000 145.000000 130.000000 115.499992
       550.000000 510.000000 470.000000 430.000000 390.999969
       925.000000 860.000000 795.000000 730.000000 666.499939
       1300.000000 1210.000000 1120.000000 1030.000000 941.999939
       1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
25 2x2 1.000000 3.000000 4.000000 6.100000
26 5x5 175.000000 160.000000 145.000000 130.000000 115.499992
       550.000000 510.000000 470.000000 430.000000 390.999969
       925.000000 860.000000 795.000000 730.000000 666.499939
       1300.000000 1210.000000 1120.000000 1030.000000 941.999939
       1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
27 2x2 1.000000 3.000000 4.000000 6.100000
28 5x5 175.000000 160.000000 145.000000 130.000000 115.499992
       550.000000 510.000000 470.000000 430.000000 390.999969
       925.000000 860.000000 795.000000 730.000000 666.499939
       1300.000000 1210.000000 1120.000000 1030.000000 941.999939
       1675.000000 1560.000000 1445.000000 1330.000000 1217.499878
```

5.7. build/tp1 <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:

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

```
1 $ build/tp1 < data/in2.txt
2 Las propiedades de multiplicacion de matrices no estan satisfechas.
```

5.8. build/tp1 <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:

```
1 12xx 12351
```

```
$ build/tp1 < data/in3.txt
Error al leer los valores de NxM en la linea 0.
```

5.9. build/tp1 <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 y el resultado de la ejecución:

```
2x2 1.1 2.2 3.3
```

```
$ build/tp1 < data/in4.txt
Faltan valores en la matriz de 2x2 de la linea 0.
```

5.10. build/tp1 <data/in7.txt

Se realiza una prueba con un archivo que contiene matrices con letras en vez de números, es decir, el formato de dato no es el esperado. Se muestra el archivo de entrada y el resultado de la ejecución:

```
2x2 1 2 3 4
2x2 1 d 3 4
```

```
./build/tp1 < data/in7.txt
Valores invalidos en la matriz de 2x2 de la linea 1.
```

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/tp1: invalid option -- 'u'
Usage:
  ./tp1 -h
  ./tp1 -V
  ./tp1  < in_file > out_file
Options:
  -V, --version Print version and quit.
  -h, --help  Print this information and quit.
Examples:
  ./tp1 < in.txt > out.txt
  cat in.txt | ./tp0 > out.txt
```

7. Conclusiones

Con el desarrollo del presente trabajo práctico pusimos en práctica la codificación en assembly MIPS y la comprensión de la conformación del stack para la función implementada. Lo que nos resultó de mayor complejidad fue el manejo de datos de tipo float. Nos encontramos también con problemas para el acceso y modificación de las matrices, pero logramos corregirlos ya que se debíeron a un error en la instrucción utilizada para manejo de array.

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 #include "argParser.h"
2
3 int showMenuHelp()
4 {
5     printf("Usage:\n");
6     printf("  ./tp1 -h\n");
7     printf("  ./tp1 -V\n");
8     printf("  ./tp1  < in_file > out_file\n");
9     printf("Options:\n");
10     printf("  -V, --version\tPrint version and quit.\n");
11     printf("  -h, --help\tPrint this information and quit.\n");
12     printf("Examples:\n");
13     printf("  ./tp1 < in.txt > out.txt\n");
14     printf("  cat in.txt | ./tp1 > out.txt\n");
15     return EXITO;
16 }
17
18 int showMenuVersion()
19 {
20   printf("\n\tTP1 - assembly MIPS\n");
21   printf("  (66.20) Organizacion de las Computadoras\n");
22   printf("-----------------------------------------------------\n");
23   printf("\t2do Cuatrimestre de 2015\n");
24   printf("\t      Version: 1.0\n");
25   printf("\nAutores:\n");
26   printf("          Opromolla, Giovanni - 87761\n");
27   printf("          Tapia, Jimena Soledad - 88392\n\n");
28   return EXITO;
29
30 }
31
32
```

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

Matrix.h

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <errno.h>
5 #include <time.h>
```

```c
#include <ctype.h>

#ifndef MATRIX_H
#define MATRIX_H

// Codigos de Error
#define ERROR_DE_ARGUMENTOS -1
#define ERROR_AL_ABRIR_ARCHIVO -2
#define MAX_BUFFER_READ 400
#define EXITO 0
#define ERROR 1

typedef struct {
  int rows;
  int columns;
  float* values;
} T_Matrix;

int doX(char *filename);
int leerEntradaEstandard();
int isDigit(char* str);
void imprimirMatriz(T_Matrix* matrix);
void liberarMatriz(T_Matrix* matrix);

T_Matrix* deserializeMatrix(const char* linea, int num_linea);
T_Matrix* procesarMatrices(T_Matrix* m1, T_Matrix* m2);
T_Matrix* nuevaMatriz(int rows, int columns);

#endif
```

Matrix.c

```c
#include "Matrix.h"
#include "MultiplyMatrix.h"
#define BUF_SIZE 255

int leerEntradaEstandard()
{

    char line[BUF_SIZE];
    int num_linea = 0;

    T_Matrix* m1 = NULL;
    T_Matrix* m2 = NULL;
    T_Matrix* m_resultado = NULL;
    /** Lectura de stdin para obtener las matrices linea por linea
        **/

    while((fgets(line,BUF_SIZE,stdin)) != NULL){

      if(strlen(line) > 2){
      if (num_linea%2 == 0)    /** Pares **/
      {
        m1 = deserializeMatrix(line, num_linea);
        if (m1 == NULL) {
          /** Algun problema surgio al deserealizar la matriz **/
          fprintf(stderr, "Ocurrio un error al procesar una las
              lineas. Verique el formato y la cantidad de matrices
              en el archivo.\n");
          break;
        }
      }
```

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

```
88    }
89
90    return res;
91  }
92
93  T_Matrix* deserializeMatrix(const char* linea, int num_linea){
94
95    int rows = -1;
96    int columns = -1;
97    int val = sscanf(linea, "%dx%d", &rows, &columns);
98
99    /** Verificar que se hayan podido leer ambos valores **/
100   if (val != 2)
101   {
102     fprintf(stderr, "Error al leer los valores de NxM en la linea %
          d.\n", num_linea);
103     exit(1);
104   }
105   /** Verificar que ambos valores sean mayores a cero **/
106   if((rows < 0) || (columns < 0)){
107     fprintf(stderr, "Los valres de NxM deben ser valores positivos.
           Linea conflictiva: %d.\n", num_linea);
108     exit(1);
109   }
110
111   /** Creo matriz y aloco su memoria **/
112   T_Matrix* matrix = nuevaMatriz(rows, columns);
113
114   /** Obtengo los valores de la matriz **/
115     char* valores = strstr (linea," ");;
116     char *token;
117     token = strtok(valores, " ");
118
119
120     int f=0, c=0;
121     while( (token != NULL) && (f < rows ) )
122     {
123       if( isDigit(token) == EXITO){
124         matrix->values[(f*columns)+c] = atof(token);
125         token = strtok(NULL, " ");
126         if(f < rows){
127           c++;
128           if(c == columns){
129             f++;
130             c = 0;
131           }
132         }
133       }
134       else{
135           fprintf(stderr, "Valores invalidos en la matriz de %dx%d
                de la linea %d.\n", matrix->rows, matrix->columns,
                num_linea );
136           liberarMatriz(matrix);
137           exit(1);
138       }
139     }
140
141     /** Matriz incompleta **/
142     if(f != rows){
143       fprintf(stderr, "Faltan valores en la matriz de %dx%d de la
            linea %d.\n", matrix->rows, matrix->columns, num_linea );
144       liberarMatriz(matrix);
```

```
145         exit(1);
146       }
147
148       /** Imprimo la matriz **/
149       //imprimirMatriz(matrix);
150
151     return matrix;
152 }
153
154 T_Matrix* nuevaMatriz(int rows, int columns){
155
156     T_Matrix* matrix = (T_Matrix*) malloc(sizeof (T_Matrix) );
157
158     matrix->rows = rows;
159     matrix->columns = columns;
160
161     /** Aloco espacio para la matriz **/
162     matrix->values = (float*) malloc (sizeof(float)*matrix->rows*
          matrix->columns);
163
164     return matrix;
165 }
166
167 T_Matrix* procesarMatrices(T_Matrix* m1, T_Matrix* m2){
168
169     T_Matrix* matrix = NULL;
170
171     if(m1->columns == m2->rows) {
172       /** Creo matriz y aloco su memoria **/
173       matrix = nuevaMatriz(m1->rows, m2->columns);
174
175       multiplyMatrixes(m1->rows, m1->columns, m2->rows, m2->columns,
            m1->values, m2->values, matrix->values);
176
177     }else{
178         fprintf(stderr, "Las propiedades de multiplicacion de
              matrices no estan satisfechas.\n");
179         exit(1);
180     }
181
182     return matrix;
183
184 }
```

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 }
```

MultiplyMatrixes.h

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <errno.h>
5  #include <time.h>
6
7  #ifndef MULTIPLY_MATRIX_H
8  #define MULTIPLY_MATRIX_H
9
10 void multiplyMatrixes(int filasM1, int columnasM1, int filasM2, int
       columnasM2, float* valoresM1, float* valoresM2, float*
      valoresMR);
11
12 #endif
```

8.2. Código MIPS

MultiplyMatrixes.S

```
1  #include <mips/regdef.h>
2  #include <sys/syscall.h>
3
4  #define SF 24
5  #define FP 20
6  #define GP 16
7  #define fp $30
8
9  #define COL_MAT_1 24
10 #define COL_MAT_2 28
11 #define ROW_MAT_1 32
12 #define ROW_MAT_2 36
13 #define VAL_MAT_1 40
14 #define VAL_MAT_2 44
15 #define VAL_MAT_RES 48
```

```
16 #define ROW1 12
17 #define COL2 8
18 #define K 4
19 #define SUM 0
20
21 .text
22 .align 2
23
24
25 .globl  multiplyMatrixes
26 .ent    multiplyMatrixes
27
28 multiplyMatrixes:
29         .frame  fp,SF,ra
30         .set    noreorder
31         .cpload t9
32         .set    reorder
33
34         #stack frame creation
35         subu    sp,sp,SF
36
37         .cprestore GP
38         sw fp, FP(sp)
39         move fp, sp
40
41         #Guardo los argumentos
42         sw a0, ROW_MAT_1(fp)        # FilasM1
43         sw a1, COL_MAT_1(fp)        # ColumnasM1
44         sw a2, ROW_MAT_2(fp)        # FilasM2
45         sw a3, COL_MAT_2(fp)        # ColumnasM2
46       #    sw t7, VAL_MAT_1(fp)       # ValuesM1
47     #     sw t8, VAL_MAT_2(fp)        # ValuesM2
48     #     sw t9, VAL_MAT_RES(fp)      # ValuesMR
49
50     sw zero, ROW1(fp)      # row1 = 0
51     lw t0, ROW_MAT_1(fp)     # t0 = filasM1
52     lw t1, ROW1(fp)      # t1 = row1
53
54 FOR_ROWS:                  # for(row1=0# row1<filasM1# ++row1)
55     beq    t1, t0, EXIT_FOR_ROWS   # if row1 == filas1 we are done
56     sw     zero, COL2(fp)       # col2 = 0
57     lw     t2, COL_MAT_2(fp)    # t2 = ColumnasM2
58     lw     t3, COL2(fp)       # t3 = column2
59
60 FOR_COLS:
61     beq    t3, t2, EXIT_FOR_COLS # for(column2=0# column2<columnasM2
           # ++column2)
62     sw     zero, SUM(fp)      # sum = 0
63     l.s    $f4, SUM(fp)
64     sw     zero, K(fp)        # k=0
65     lw     t4, COL_MAT_1(fp)    # t4 = ColumnasM1
66     lw     t5, K(fp)        # t5 = k
67     #j EXIT_FOR_SUM
68 FOR_SUM:
69     beq    t5, t4, EXIT_FOR_SUM  # for(k=0# k<columnasM1# k++)
70     #valoresM1[(row1*columnasM1) + k]
71     lw   t6, VAL_MAT_1(fp )     # valoresM1
72     mul    t7, t1, t4          # t7 = row1*columnasM1
73     addu   t7, t7, t5          # t7 = (row1*columnasM1) + k
74     sll    t7, t7, 2         # Aligned t7 * 4
75     addu   t6, t6, t7          # valoresM1[(row1*columnasM1) + k]
76     l.s      $f0, 0(t6)          # f0 = valoresM1[(row1*columnasM1) +
```

```
        k]
77      #valoresM2[(k *columnasM2) + column2]
78      lw   t6, VAL_MAT_2(fp)      # valoresM2
79      mul    t7, t5, t2          # t7 = k *columnasM2
80      addu   t7, t7, t3          # (k *columnasM2) + column2
81      sll    t7, t7, 2         # Aligned t7 * 4
82      addu   t6, t6, t7          # valoresM2[(k *columnasM2) + column2]
83      l.s      $f2, 0(t6)            # f1 = valoresM2[(k *columnasM2) +
        column2]
84      #sum=sum + valoresM1[(row1*columnasM1) + k] * valoresM2[(k *
        columnasM2) + column2]#
85      mul.s $f6, $f0, $f2         # valoresM1[(row1*columnasM1) + k] *
        valoresM2[(k *columnasM2) + column2]
86      add.s $f4, $f4, $f6         # sum=sum + valoresM1[(row1*
        columnasM1) + k] * valoresM2[(k *columnasM2) + column2]#
87      s.s    $f4, SUM(fp)         #
88      addi   t5, t5, 1           # add 1 to t5 (k)
89      j FOR_SUM             # jump back to the top
90
91  EXIT_FOR_SUM:
92      #valoresMR[(row1*columnasM2) + column2]=sum#
93      lw     t6, VAL_MAT_RES(fp)
94      #addu t6, fp, VAL_MAT_RES   # valoresMR
95      mul    t7, t1, t2          # t7 = row1*columnasM2
96      addu   t7, t7, t3          # t7 = (row1*columnasM2) + column2
97      sll    t7, t7, 2         # Aligned t7 * 4
98      addu   t6, t6, t7          # t6 = valoresMR[(row1*columnasM2) +
        column2]
99      l.s    $f4, SUM(fp)          # f4 = sum
100     s.s    $f4, 0(t6)            # valoresMR[(row1*columnasM2) +
        column2] = sum
101     addi   t3, t3, 1           # add 1 to t3 (column2)
102     j FOR_COLS             # jump back to the top
103
104 EXIT_FOR_COLS:
105     addi   t1, t1, 1           # add 1 to t1 (row1)
106     j FOR_ROWS             # jump back to the top
107
108 EXIT_FOR_ROWS:
109     move   sp, fp
110     lw   fp, FP(sp)
111     lw   gp, GP(sp)
112
113     addu   sp, sp, SF
114     j ra
115
116 .end multiplyMatrixes
```

Referencias

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

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