

Universidad de Buenos Aires

Facultad De Ingeniería

Año 2016 - 1er Cuatrimestre

**66.20 Organización de Computadoras - Práctica Martes**

TRABAJO PRÁCTICO Nº 0

FECHA: 29/03/2016

INTEGRANTES:

Juan Manuel Zaragoza - #92308

<juanmanuelzar@gmail.com>

Nicolás Truksinas- #88452

<nicotruk@gmail.com>

Santiago Weber- #93789

<santiago.weber91@gmail.com>

1. **Objetivo**

Este trabajo práctico tiene como objetivo demostrar que los alumnos tienen los conocimientos y herramientas necesarios para elaborar los futuros trabajos prácticos.

1. **Resumen**

En este trabajo práctico se implementó un simple programa en lenguaje C, en el que se multiplican matrices de n x n. Luego se compiló este programa utilizando el compilador gcc, para generar el código assembly correspondiente, en arquitectura mips 32. Para ello se emuló un sistema operativo, NetBSD, utilizando gxemul, sobre el sistema operativo anfitrión (Linux).

1. **Diseño**
   1. **Tipos de datos utilizados**

/\* Estructura de matriz \*/

typedef struct matrix {

size\_t rows;

size\_t cols;

float\* array;

} matrix\_t;

/\* Enumerados \*/

typedef enum {

PRINT\_HELP, PRINT\_VERSION

} message\_t;

typedef enum {

SUCCESS, ERROR, CONTINUE

**} status\_t;**

* 1. **Funciones**

**matrix\_t \*create\_matrix(size\_t rows, size\_t cols):** Constructuye una matrix\_t pidiendo la memoria suficiente para una matriz de rows\*cols.

**void destroy\_matrix(matrix\_t \*m):** Destruye una matrix\_t liberando la memoria pedida al momento de construirla-

**int print\_matrix(FILE\* fp, matrix\_t\* m):** Imprime matrix\_t, iterando cada uno de sus elementos, sobre el file pointer fp, en este caso, stdout.

**matrix\_t\* matrix\_multiply(matrix\_t\* m1, matrix\_t\* m2):** Multiplica las matrices m1 y m2. Para esto se fijan los índices j e i en el array de resultados y se van recorriendo las filas de la primer matriz (fijando la columna) y las columnas de la segunda (fijando las filas).

**status\_t handleParam(int argc, char\* argv[], FILE \*\*fp\_in):** Valida y guarda los parámetros de entrada. En caso de los modificadores “-h” o “-V” se imprimen mensajes de ayuda, sino se abre el archivo explicitado como argumento.

**void print\_message(message\_t message):** Imprime los mensajes de ayuda y de versión.

1. **Uso y errores**
   1. **Modo de uso**

Ayuda:

tp0 -h

Versión:

tp0 -V

Entrada y salida con archivos:

tp0 < in\_file > out\_file

cat in.txt | tp0 > out.txt

* 1. **Mensajes y errores**

En caso de que los parámetros de entrada no sean los correctos (distintos a “-h” o “-V”) el programa devolverá el mensaje **Error al procesar los parámetros de entrada**.

En caso de que el archivo de entrada no esté formado como el esperado el programa devolverá el mensaje **Mal formato de linea**.

1. **Pruebas**
   1. **Prueba hasta n = 4**

Entrada:

2 1 2 3 4 5 6 2 8

3 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3 1.4 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3 1.4

4 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3 1.4 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3 1.4 1.2 1.3 1.4 1.5 1.3 1.3 1.3

Salida:

2 9 22 23 50

3 5.21 5.07 5.33 5.44 5.33 5.61 5.33 5.2 5.47

4 7.34 6.88 7.14 7.4 7.02 6.63 6.89 7.15 7.16 6.76 7.03 7.3 7.26 6.89 7.17 7.45

* 1. **Mal formato de línea: Un número de más**

Entrada:

2 1 2 3 4 5 6 2 8 2

3 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3 1.4 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3 1.4

Salida:

Mal formato de linea.

* 1. **Mal formato de línea: Un número de menos**

Entrada:

2 1 2 3 4 5 6 2 8

3 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3 1.4 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3

4 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3 1.4 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3 1.4 1.2 1.3 1.4 1.5 1.3 1.3 1.3

Salida:

2 9 22 23 50

Mal formato de linea.

* 1. **Mal formato de línea: letra en una fila**

Entrada:

2 1 2 3 4 5 6 a 8

3 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3 1.4 1.2 1.3 1.4 1.5 1.3 1.3 1.3 1.3 1.4

Salida:

Mal formato de linea.

**7. Conclusiones**

Por medio de un emulador como el gxemul, podemos utilizar un sistema operativo invitado, montado sobre el sistema operativo anfitrión, y así generar código assembly para una arquitectura distinta a la del ordenador en el cuál se está ejecutando el programa, en este caso, MIPS 32.

**8. Código C**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\* Constantes \*/

const char \*cmdmsg[] = {

"Usage:\n"

" tp0 -h\n"

" tp0 -V\n"

" tp0 < in\_file > out\_file\n"

"Options:\n"

" -V, --version Print version and quit.\n"

" -h, --help Print this information and quit.\n"

"Examples:\n"

" tp0 < in.txt > out.txt\n"

" cat in.txt | tp0 > out.txt\n",

"tp0 v0\n"

};

/\* Estructura de matriz \*/

typedef struct matrix {

size\_t rows;

size\_t cols;

float\* array;

} matrix\_t;

/\* Enumerados \*/

typedef enum {

PRINT\_HELP, PRINT\_VERSION

} message\_t;

typedef enum {

SUCCESS, ERROR, CONTINUE

} status\_t;

/\* Prototipos \*/

status\_t handleParam(int, char \*[], FILE \*\*);

void print\_message(message\_t);

matrix\_t\* create\_matrix(size\_t, size\_t);

void destroy\_matrix(matrix\_t\*);

int print\_matrix(FILE\*, matrix\_t\*);

matrix\_t\* matrix\_multiply(matrix\_t\*, matrix\_t\*);

int main(int argc, char\* argv[]) {

matrix\_t \*matrix1, \*matrix2, \*matrix\_result;

status\_t status;

FILE \*fp\_in;

char ch;

int num\_rows = 0, i = 0;

float element;

fp\_in = stdin;

status = handleParam(argc, argv, &fp\_in);

if (status == SUCCESS) {

return EXIT\_SUCCESS;

} else if (status == ERROR) {

fprintf(stderr, "%s\n", "Error al procesar los parámetros de entrada");

return EXIT\_FAILURE;

} else {

while (!feof(fp\_in)) {

if (fscanf(fp\_in, "%d", &num\_rows) == 1) {

matrix1 = create\_matrix(num\_rows, num\_rows);

matrix2 = create\_matrix(num\_rows, num\_rows);

i = 0;

while (i < num\_rows \* num\_rows

&& fscanf(fp\_in, "%g", &element) == 1) {

matrix1->array[i] = element;

i++;

}

while (i >= (num\_rows \* num\_rows)

&& (i < num\_rows \* num\_rows \* 2)

&& fscanf(fp\_in, "%g", &element) == 1) {

matrix2->array[i - num\_rows \* num\_rows] = element;

i++;

}

if (fscanf(fp\_in, "%c", &ch) == 1 && ch != '\n') {

fprintf(stderr, "%s\n", "Mal formato de linea.");

return EXIT\_FAILURE;

}

matrix\_result = matrix\_multiply(matrix1, matrix2);

print\_matrix(stdout, matrix\_result);

destroy\_matrix(matrix1);

destroy\_matrix(matrix2);

destroy\_matrix(matrix\_result);

}

}

}

fclose(fp\_in);

return EXIT\_SUCCESS;

}

status\_t handleParam(int argc, char\* argv[], FILE \*\*fp\_in) {

size\_t i;

FILE \*f;

for (i = 1; i < argc; i++) {

if (strcmp(argv[i], "-h") == 0) {

print\_message(PRINT\_HELP);

return SUCCESS;

} else if (strcmp(argv[i], "-V") == 0) {

print\_message(PRINT\_VERSION);

return SUCCESS;

} else {

f = fopen(argv[i], "r");

if (f != NULL) {

\*fp\_in = f;

return CONTINUE;

}

}

}

return CONTINUE;

}

/\* Imprimir mensajes \*/

void print\_message(message\_t message) {

printf("%s", cmdmsg[message]);

}

/\* Construir matriz \*/

matrix\_t \*create\_matrix(size\_t rows, size\_t cols) {

matrix\_t \*matrix = (matrix\_t \*) calloc(1, sizeof(matrix\_t));

matrix->rows = rows;

matrix->cols = cols;

matrix->array = (float \*) malloc((rows \* cols) \* sizeof(float));

return matrix;

}

/\* Destruir matriz \*/

void destroy\_matrix(matrix\_t \*m) {

free(m->array);

free(m);

}

/\* Imprimir matriz \*/

int print\_matrix(FILE\* fp, matrix\_t\* m) {

size\_t i;

fprintf(fp, "%zu ", m->rows);

for (i = 0; i < (m->rows) \* (m->cols); i++) {

fprintf(fp, "%g ", m->array[i]);

}

fprintf(fp, "\n");

return i;

}

/\* Multiplica dos matrices \*/

matrix\_t\* matrix\_multiply(matrix\_t\* m1, matrix\_t\* m2) {

size\_t i, j, k;

matrix\_t \*m\_res = create\_matrix(m1->rows, m2->cols);

for (i = 0; i < m1->rows; i++) {

for (j = 0; j < m2->cols; j++) {

/\* con j recorro la fila y con i voy saltando de columnas \*/

m\_res->array[j + i \* m2->cols] = 0;

for (k = 0; k < m1->cols; k++) {/\* recorro la fila de m1 y la columna de m2 \*/

m\_res->array[j + i \* m2->cols] += m1->array[k + i \* m1->cols]

\* m2->array[j + k \* m2->cols];

}

/\* m1->array[k + i\*m1->cols] fijo la fila y recorro las columnas \*/

/\* m2->array[j + k\*m2->cols] fijo la columna y recorro las filas \*/

}

}

return m\_res;

}

**9. Código assembly MIPS 32**

.file 1 "tp0.c"

.section .mdebug.abi32

.previous

.abicalls

.rdata

.align 2

$LC0:

.ascii "Usage:\n"

.ascii " tp0 -h\n"

.ascii " tp0 -V\n"

.ascii " tp0 < in\_file > out\_file\n"

.ascii "Options:\n"

.ascii " -V, --version\tPrint version and quit.\n"

.ascii " -h, --help\tPrint this information and quit.\n"

.ascii "Examples:\n"

.ascii " tp0 < in.txt > out.txt\n"

.ascii " cat in.txt | tp0 > out.txt\n\000"

.align 2

$LC1:

.ascii "tp0 v0\n\000"

.globl cmdmsg

.data

.align 2

.type cmdmsg, @object

.size cmdmsg, 8

cmdmsg:

.word $LC0

.word $LC1

.rdata

.align 2

$LC2:

.ascii "%s\n\000"

.align 2

$LC3:

.ascii "Error al procesar los par\303\241metros de entrada\000"

.align 2

$LC4:

.ascii "%d\000"

.align 2

$LC5:

.ascii "%g\000"

.align 2

$LC6:

.ascii "%c\000"

.align 2

$LC7:

.ascii "Mal formato de linea.\000"

.text

.align 2

.globl main

.ent main

main:

.frame $fp,80,$ra # vars= 40, regs= 3/0, args= 16, extra= 8

.mask 0xd0000000,-8

.fmask 0x00000000,0

.set noreorder

.cpload $t9

.set reorder

subu $sp,$sp,80

.cprestore 16

sw $ra,72($sp)

sw $fp,68($sp)

sw $gp,64($sp)

move $fp,$sp

sw $a0,80($fp)

sw $a1,84($fp)

sw $zero,48($fp)

sw $zero,52($fp)

la $v0,\_\_sF

sw $v0,40($fp)

addu $v0,$fp,40

lw $a0,80($fp)

lw $a1,84($fp)

move $a2,$v0

la $t9,handleParam

jal $ra,$t9

sw $v0,36($fp)

lw $v0,36($fp)

bne $v0,$zero,$L18

sw $zero,60($fp)

b $L17

$L18:

lw $v1,36($fp)

li $v0,1 # 0x1

bne $v1,$v0,$L22

la $a0,\_\_sF+176

la $a1,$LC2

la $a2,$LC3

la $t9,fprintf

jal $ra,$t9

li $v0,1 # 0x1

sw $v0,60($fp)

b $L17

$L22:

lw $v0,40($fp)

lhu $v0,12($v0)

srl $v0,$v0,5

andi $v0,$v0,0x1

beq $v0,$zero,$L24

b $L19

$L24:

addu $v0,$fp,48

lw $a0,40($fp)

la $a1,$LC4

move $a2,$v0

la $t9,fscanf

jal $ra,$t9

move $v1,$v0

li $v0,1 # 0x1

bne $v1,$v0,$L22

lw $a0,48($fp)

lw $a1,48($fp)

la $t9,create\_matrix

jal $ra,$t9

sw $v0,24($fp)

lw $a0,48($fp)

lw $a1,48($fp)

la $t9,create\_matrix

jal $ra,$t9

sw $v0,28($fp)

sw $zero,52($fp)

$L26:

lw $v1,48($fp)

lw $v0,48($fp)

mult $v1,$v0

mflo $v1

lw $v0,52($fp)

slt $v0,$v0,$v1

beq $v0,$zero,$L27

addu $v0,$fp,56

lw $a0,40($fp)

la $a1,$LC5

move $a2,$v0

la $t9,fscanf

jal $ra,$t9

move $v1,$v0

li $v0,1 # 0x1

beq $v1,$v0,$L28

b $L27

$L28:

lw $a0,24($fp)

lw $v0,52($fp)

sll $v1,$v0,2

lw $v0,8($a0)

addu $v0,$v1,$v0

l.s $f0,56($fp)

s.s $f0,0($v0)

lw $v0,52($fp)

addu $v0,$v0,1

sw $v0,52($fp)

b $L26

$L27:

.set noreorder

nop

.set reorder

$L30:

lw $v1,48($fp)

lw $v0,48($fp)

mult $v1,$v0

mflo $v1

lw $v0,52($fp)

slt $v0,$v0,$v1

bne $v0,$zero,$L31

lw $v1,48($fp)

lw $v0,48($fp)

mult $v1,$v0

mflo $v0

sll $v1,$v0,1

lw $v0,52($fp)

slt $v0,$v0,$v1

beq $v0,$zero,$L31

addu $v0,$fp,56

lw $a0,40($fp)

la $a1,$LC5

move $a2,$v0

la $t9,fscanf

jal $ra,$t9

move $v1,$v0

li $v0,1 # 0x1

beq $v1,$v0,$L32

b $L31

$L32:

lw $a0,28($fp)

lw $v1,48($fp)

lw $v0,48($fp)

mult $v1,$v0

mflo $v1

lw $v0,52($fp)

subu $v0,$v0,$v1

sll $v1,$v0,2

lw $v0,8($a0)

addu $v0,$v1,$v0

l.s $f0,56($fp)

s.s $f0,0($v0)

lw $v0,52($fp)

addu $v0,$v0,1

sw $v0,52($fp)

b $L30

$L31:

addu $v0,$fp,44

lw $a0,40($fp)

la $a1,$LC6

move $a2,$v0

la $t9,fscanf

jal $ra,$t9

move $v1,$v0

li $v0,1 # 0x1

bne $v1,$v0,$L34

lb $v1,44($fp)

li $v0,10 # 0xa

beq $v1,$v0,$L34

la $a0,\_\_sF+176

la $a1,$LC2

la $a2,$LC7

la $t9,fprintf

jal $ra,$t9

li $v0,1 # 0x1

sw $v0,60($fp)

b $L17

$L34:

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 $a0,24($fp)

la $t9,destroy\_matrix

jal $ra,$t9

lw $a0,28($fp)

la $t9,destroy\_matrix

jal $ra,$t9

lw $a0,32($fp)

la $t9,destroy\_matrix

jal $ra,$t9

b $L22

$L19:

lw $a0,40($fp)

la $t9,fclose

jal $ra,$t9

sw $zero,60($fp)

$L17:

lw $v0,60($fp)

move $sp,$fp

lw $ra,72($sp)

lw $fp,68($sp)

addu $sp,$sp,80

j $ra

.end main

.size main, .-main

.rdata

.align 2

$LC8:

.ascii "-h\000"

.align 2

$LC9:

.ascii "-V\000"

.align 2

$LC10:

.ascii "r\000"

.text

.align 2

.globl handleParam

.ent handleParam

handleParam:

.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 $a2,64($fp)

li $v0,1 # 0x1

sw $v0,24($fp)

$L36:

lw $v0,24($fp)

lw $v1,56($fp)

sltu $v0,$v0,$v1

bne $v0,$zero,$L39

b $L37

$L39:

lw $v0,24($fp)

sll $v1,$v0,2

lw $v0,60($fp)

addu $v0,$v1,$v0

lw $a0,0($v0)

la $a1,$LC8

la $t9,strcmp

jal $ra,$t9

bne $v0,$zero,$L40

move $a0,$zero

la $t9,print\_message

jal $ra,$t9

sw $zero,32($fp)

b $L35

$L40:

lw $v0,24($fp)

sll $v1,$v0,2

lw $v0,60($fp)

addu $v0,$v1,$v0

lw $a0,0($v0)

la $a1,$LC9

la $t9,strcmp

jal $ra,$t9

bne $v0,$zero,$L42

li $a0,1 # 0x1

la $t9,print\_message

jal $ra,$t9

sw $zero,32($fp)

b $L35

$L42:

lw $v0,24($fp)

sll $v1,$v0,2

lw $v0,60($fp)

addu $v0,$v1,$v0

lw $a0,0($v0)

la $a1,$LC10

la $t9,fopen

jal $ra,$t9

sw $v0,28($fp)

lw $v0,28($fp)

beq $v0,$zero,$L38

lw $v1,64($fp)

lw $v0,28($fp)

sw $v0,0($v1)

li $v0,2 # 0x2

sw $v0,32($fp)

b $L35

$L38:

lw $v0,24($fp)

addu $v0,$v0,1

sw $v0,24($fp)

b $L36

$L37:

li $v0,2 # 0x2

sw $v0,32($fp)

$L35:

lw $v0,32($fp)

move $sp,$fp

lw $ra,48($sp)

lw $fp,44($sp)

addu $sp,$sp,56

j $ra

.end handleParam

.size handleParam, .-handleParam

.rdata

.align 2

$LC11:

.ascii "%s\000"

.text

.align 2

.globl print\_message

.ent print\_message

print\_message:

.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)

sll $v1,$v0,2

la $v0,cmdmsg

addu $v0,$v1,$v0

la $a0,$LC11

lw $a1,0($v0)

la $t9,printf

jal $ra,$t9

move $sp,$fp

lw $ra,32($sp)

lw $fp,28($sp)

addu $sp,$sp,40

j $ra

.end print\_message

.size print\_message, .-print\_message

.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 $a0,1 # 0x1

li $a1,12 # 0xc

la $t9,calloc

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 $v0,$v0,2

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 $a0,40($fp)

la $t9,free

jal $ra,$t9

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

$LC12:

.ascii "%zu \000"

.align 2

$LC13:

.ascii "%g \000"

.align 2

$LC14:

.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)

lw $v0,52($fp)

lw $a0,48($fp)

la $a1,$LC12

lw $a2,0($v0)

la $t9,fprintf

jal $ra,$t9

sw $zero,24($fp)

$L49:

lw $v0,52($fp)

lw $v1,52($fp)

lw $a0,0($v0)

lw $v0,4($v1)

mult $a0,$v0

mflo $v1

lw $v0,24($fp)

sltu $v0,$v0,$v1

bne $v0,$zero,$L52

b $L50

$L52:

lw $a0,52($fp)

lw $v0,24($fp)

sll $v1,$v0,2

lw $v0,8($a0)

addu $v0,$v1,$v0

l.s $f0,0($v0)

cvt.d.s $f0,$f0

lw $a0,48($fp)

la $a1,$LC13

mfc1 $a2,$f0

mfc1 $a3,$f1

la $t9,fprintf

jal $ra,$t9

lw $v0,24($fp)

addu $v0,$v0,1

sw $v0,24($fp)

b $L49

$L50:

lw $a0,48($fp)

la $a1,$LC14

la $t9,fprintf

jal $ra,$t9

lw $v0,24($fp)

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 $fp,56,$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)

lw $v0,56($fp)

lw $v1,60($fp)

lw $a0,0($v0)

lw $a1,4($v1)

la $t9,create\_matrix

jal $ra,$t9

sw $v0,36($fp)

sw $zero,24($fp)

$L54:

lw $v0,56($fp)

lw $v1,24($fp)

lw $v0,0($v0)

sltu $v0,$v1,$v0

bne $v0,$zero,$L57

b $L55

$L57:

sw $zero,28($fp)

$L58:

lw $v0,60($fp)

lw $v1,28($fp)

lw $v0,4($v0)

sltu $v0,$v1,$v0

bne $v0,$zero,$L61

b $L56

$L61:

lw $a0,36($fp)

lw $v0,60($fp)

lw $v1,24($fp)

lw $v0,4($v0)

mult $v1,$v0

mflo $v1

lw $v0,28($fp)

addu $v0,$v1,$v0

sll $v1,$v0,2

lw $v0,8($a0)

addu $v0,$v1,$v0

sw $zero,0($v0)

sw $zero,32($fp)

$L62:

lw $v0,56($fp)

lw $v1,32($fp)

lw $v0,4($v0)

sltu $v0,$v1,$v0

bne $v0,$zero,$L65

b $L60

$L65:

lw $a0,36($fp)

lw $v0,60($fp)

lw $v1,24($fp)

lw $v0,4($v0)

mult $v1,$v0

mflo $v1

lw $v0,28($fp)

addu $v0,$v1,$v0

sll $v1,$v0,2

lw $v0,8($a0)

addu $a2,$v1,$v0

lw $a0,36($fp)

lw $v0,60($fp)

lw $v1,24($fp)

lw $v0,4($v0)

mult $v1,$v0

mflo $v1

lw $v0,28($fp)

addu $v0,$v1,$v0

sll $v1,$v0,2

lw $v0,8($a0)

addu $a3,$v1,$v0

lw $a0,56($fp)

lw $v0,56($fp)

lw $v1,24($fp)

lw $v0,4($v0)

mult $v1,$v0

mflo $v1

lw $v0,32($fp)

addu $v0,$v1,$v0

sll $v1,$v0,2

lw $v0,8($a0)

addu $a1,$v1,$v0

lw $a0,60($fp)

lw $v0,60($fp)

lw $v1,32($fp)

lw $v0,4($v0)

mult $v1,$v0

mflo $v1

lw $v0,28($fp)

addu $v0,$v1,$v0

sll $v1,$v0,2

lw $v0,8($a0)

addu $v0,$v1,$v0

l.s $f2,0($a1)

l.s $f0,0($v0)

mul.s $f2,$f2,$f0

l.s $f0,0($a3)

add.s $f0,$f0,$f2

s.s $f0,0($a2)

lw $v0,32($fp)

addu $v0,$v0,1

sw $v0,32($fp)

b $L62

$L60:

lw $v0,28($fp)

addu $v0,$v0,1

sw $v0,28($fp)

b $L58

$L56:

lw $v0,24($fp)

addu $v0,$v0,1

sw $v0,24($fp)

b $L54

$L55:

lw $v0,36($fp)

move $sp,$fp

lw $ra,48($sp)

lw $fp,44($sp)

addu $sp,$sp,56

j $ra

.end matrix\_multiply

.size matrix\_multiply, .-matrix\_multiply

.ident "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"