FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

66.20 Organización de Computadoras
Primer Cuatrimestre del 2016

Trabajo Práctico 0: **Infraestructura básica**

Integrantes

de la Fuente, Hernán - Padrón: 95730
López Pecora, Juan Ignacio - Padrón: 84700

Índice general

Introducción

Este trabajo práctico es el inicial dentro del curso de la materia [66.20] Organización de Computadoras, de la Facultad de Ingeniería de la Universidad de Buenos Aires. Tiene por objetivo resolver un algoritmo sencillo de multiplicación de matrices cargadas manualmente mediante una representación del tipo row major order y obtener su resultado.

Desarrollo

0.1. Aclaraciones

Se toma como hipótesis que el usuario ingresará sólo dos matrices de dimensiones cuadradas, por lo tanto, esta implementación solo garantiza su funcionamiento en estas condiciones.

Respecto a la organización del código fuente de la aplicación, se optó por separar lo pertinente a la matriz por un lado (*matrix.h & matrix.c*), por otro el motor o 'core' de la aplicación(*engine.h & engine.c*) y, por último, el contexto desde el cual se invoca dicho motor (*app.c*). Además, contamos con un *debug.h* que facilita la tarea de mantenimiento.

Para compilar el programa deberá correrse el siguiente comando:

gcc -Wall -O0 -o tp0 app.c engine.c matrix.c

De esta manera, se generará en la misma carpeta un binario con el nombre tp0 listo para ser ejecutado.

Por otro lado, para poder generar el código assembly, deberá correrse el mismo comando pero agregando '-S -mrnames'. Sería:

gcc -Wall -O0 -S -mrnames app.c engine.c matrix.c

0.2. Código

matrix.h

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

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

matrix_t* create_matrix(size_t rows, size_t cols);

void destroy_matrix(matrix_t *m);

int print_matrix(FILE* fp, matrix_t *m);

matrix_t* matrix_multiply(matrix_t *m1, matrix_t* m2);

void print_matrix_2d(matrix_t *m);
```

matrix.c

```
#include "matrix.h"

matrix_t* create_matrix(size_t rows, size_t cols) {
    matrix_t *m = (matrix_t*) malloc(sizeof(size_t) * 2
        + sizeof(double*));
    m->rows = rows;
    m->cols = cols;
    m->array = (double*) malloc(sizeof(double) * rows *
        cols);
    return m;
}

void destroy_matrix(matrix_t *m) {
    free(m->array);
    free(m);
}

int print_matrix(FILE* fp, matrix_t *m) {
    int i;
    fprintf(fp, "%d", (int) m->rows);
    for (i = 0; i < m->rows * m->cols; ++i) {
        fprintf(fp, " %g", m->array[i]);
    }
    fprintf(fp, "\n");
    return 0;
}

matrix_t* matrix_multiply(matrix_t *m1, matrix_t* m2) {
    matrix_t *m3 = create_matrix(m1->rows, m2->cols);
    int dim = m1->rows; // hypothesis: square matrixes
    int i, j;
    for (i = 0; i < dim * dim; ++i) {
        m3->array[i] = 0;
        for (j = 0; j < dim; ++j) {
            m3->array[i] += m1->array[(i / dim)*dim + j
               ] * m2->array[i %dim + j * dim];
        }
    }
    return m3;
}

void print_matrix_2d(matrix_t *m) {
    int i;
    for (i = 0; i < m->rows * m->cols; ++i) {
        if (i %m->cols == 0) {
            printf("\n");
        }
        printf("%f ", m->array[i]);
    }
    printf("\n");
}
```

matrix.s

```
        .file   1 "src/matrix.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .text
        .align  2
        .globl  create_matrix
        .ent    create_matrix
create_matrix:
        .frame  $fp,48,$ra              # vars= 8, regs
           = 4/0, args= 16, extra= 8
        .mask   0xd0010000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,48
        .cprestore 16
        sw      $ra,44($sp)
        sw      $fp,40($sp)
        sw      $gp,36($sp)
        sw      $s0,32($sp)
        move    $fp,$sp
        sw      $a0,48($fp)
        sw      $a1,52($fp)
        li      $a0,12                  # 0xc
        la      $t9,malloc
        jal     $ra,$t9
        sw      $v0,24($fp)
        lw      $v1,24($fp)
        lw      $v0,48($fp)
        sw      $v0,0($v1)
        lw      $v1,24($fp)
        lw      $v0,52($fp)
        sw      $v0,4($v1)
        lw      $s0,24($fp)
        lw      $v1,48($fp)
        lw      $v0,52($fp)
        mult    $v1,$v0
        mflo    $v0
        sll     $v0,$v0,3
        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
$LC0:
        .ascii  "%d\000"
        .align  2
$LC1:
        .ascii  " %g\000"
        .align  2
```

```
$LC2:
        .ascii  "\n\000"
        .text
        .align  2
        .globl  print_matrix
        .ent    print_matrix
print_matrix:
        .frame  $fp,48,$ra              # vars= 8, regs
          = 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,48
        .cprestore 16
        sw      $ra,40($sp)
        sw      $fp,36($sp)
        sw      $gp,32($sp)
        move    $fp,$sp
        sw      $a0,48($fp)
        sw      $a1,52($fp)
        lw      $v0,52($fp)
        lw      $a0,48($fp)
        la      $a1,$LC0
        lw      $a2,0($v0)
        la      $t9,fprintf
        jal     $ra,$t9
        sw      $zero,24($fp)
$L20:
        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,$L23
        b       $L21
$L23:
        lw      $a0,52($fp)
        lw      $v0,24($fp)
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $v0,$v1,$v0
        lw      $a0,48($fp)
        la      $a1,$LC1
        lw      $a2,0($v0)
        lw      $a3,4($v0)
        la      $t9,fprintf
        jal     $ra,$t9
        lw      $v0,24($fp)
        addu    $v0,$v0,1
        sw      $v0,24($fp)
        b       $L20
$L21:
        lw      $a0,48($fp)
        la      $a1,$LC2
        la      $t9,fprintf
        jal     $ra,$t9
        move    $v0,$zero
        move    $sp,$fp
        lw      $ra,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        j       $ra
        .end    print_matrix
        .size   print_matrix, .-print_matrix
        .align  2
        .globl  matrix_multiply
        .ent    matrix_multiply
matrix_multiply:
        .frame  $fp,56,$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      $v0,24($fp)
        lw      $v0,56($fp)
        lw      $v0,0($v0)
        sw      $v0,28($fp)
        sw      $zero,32($fp)
$L25:
        lw      $v1,28($fp)
        lw      $v0,28($fp)
        mult    $v1,$v0
        mflo    $v1
        lw      $v0,32($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L28
        b       $L26
$L28:
        lw      $a0,24($fp)
        lw      $v0,32($fp)
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $v0,$v1,$v0
        sw      $zero,0($v0)
        sw      $zero,4($v0)
        sw      $zero,36($fp)
$L29:
        lw      $v0,36($fp)
        lw      $v1,28($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L32
        b       $L27
```

```
$L32:
        lw      $a0,24($fp)
        lw      $v0,32($fp)
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $a3,$v1,$v0
        lw      $a0,24($fp)
        lw      $v0,32($fp)
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $t0,$v1,$v0
        lw      $a0,56($fp)
        lw      $v1,32($fp)
        lw      $v0,28($fp)
        div     $0,$v1,$v0
        mflo    $v1
        .set    noreorder
        bne     $v0,$0,1f
        nop
        break   7
1:
        .set    reorder
        lw      $v0,28($fp)
        mult    $v1,$v0
        mflo    $v1
        lw      $v0,36($fp)
        addu    $v0,$v1,$v0
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $a1,$v1,$v0
        lw      $a2,60($fp)
        lw      $v1,32($fp)
        lw      $v0,28($fp)
        div     $0,$v1,$v0
        mfhi    $a0
        .set    noreorder
        bne     $v0,$0,1f
        nop
        break   7
1:
        .set    reorder
        lw      $v1,36($fp)
        lw      $v0,28($fp)
        mult    $v1,$v0
        mflo    $v0
        addu    $v0,$a0,$v0
        sll     $v1,$v0,3
        lw      $v0,8($a2)
        addu    $v0,$v1,$v0
        l.d     $f2,0($a1)
        l.d     $f0,0($v0)
        mul.d   $f2,$f2,$f0
        l.d     $f0,0($t0)
        add.d   $f0,$f0,$f2
        s.d     $f0,0($a3)
        lw      $v0,36($fp)
        addu    $v0,$v0,1
        sw      $v0,36($fp)
        b       $L29
$L27:
        lw      $v0,32($fp)
        addu    $v0,$v0,1
        sw      $v0,32($fp)
        b       $L25
$L26:
        lw      $v0,24($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
        .rdata
        .align  2
$LC3:
        .ascii  "%f \000"
        .text
        .align  2
        .globl  print_matrix_2d
        .ent    print_matrix_2d
print_matrix_2d:
        .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       $zero,24($fp)
$L34:
        lw       $v0,48($fp)
        lw       $v1,48($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,$L37
        b        $L35
$L37:
        lw       $v0,48($fp)
        lw       $v1,24($fp)
        lw       $v0,4($v0)
        divu     $0,$v1,$v0
        mfhi     $v1
        .set     noreorder
        bne      $v0,$0,1f
        nop
        break    7
1:
        .set     reorder
        bne      $v1,$zero,$L38
        la       $a0,$LC2
        la       $t9,printf
        jal     $ra,$t9
$L38:
        lw      $a0,48($fp)
        lw      $v0,24($fp)
        sll     $v1,$v0,3
        lw      $v0,8($a0)
        addu    $v0,$v1,$v0
        la      $a0,$LC3
        lw      $a2,0($v0)
        lw      $a3,4($v0)
        la      $t9,printf
        jal     $ra,$t9
        lw      $v0,24($fp)
        addu    $v0,$v0,1
        sw      $v0,24($fp)
        b       $L34
$L35:
        la      $a0,$LC2
        la      $t9,printf
        jal     $ra,$t9
        move    $sp,$fp
        lw      $ra,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        j       $ra
        .end    print_matrix_2d
        .size   print_matrix_2d, .-print_matrix_2d
        .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)
           "
```

engine.h

```
#define ERR_INVALID_INPUT 1
#define BUFFER_SIZE       15

void run();
```

engine.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
```

```
#include <errno.h>
#include "debug.h"
#include "engine.h"
#include "matrix.h"

static int fgetline(FILE *f, char** line, int len);
static int parseline(char *filepath, matrix_t **m1,
   matrix_t **m2);
static int loadMatrix(char **line, matrix_t *m);
static int getInteger(int *d, char **linepos);
static int getDouble(double *d, char **linepos);

void run() {
     matrix_t *m1 = NULL;
     matrix_t *m2 = NULL;
     matrix_t *m3 = NULL;

     char* line;
     while (!feof(stdin)) {
          line = malloc(sizeof(char) * BUFFER_SIZE);
          if (fgetline(stdin, &line, BUFFER_SIZE) > 0) {
               debug_print("line: %s\n", line);
               if (parseline(line, &m1, &m2) != 0) {
                    free(line);
                    fprintf(stderr, "parse error\n");
                    exit(EXIT_FAILURE);
               }

               m3 = matrix_multiply(m1, m2);
               #ifdef DEBUG
               print_matrix_2d(m3);
               #endif
               print_matrix(stdout, m3);

               destroy_matrix(m1);
               destroy_matrix(m2);
               destroy_matrix(m3);

          }
          free(line);
     }
```

```
}

static int fgetline(FILE *f, char** buff, int len) {
    int i;
    char c;
    for (i = 0; i < len - 1 && (c = fgetc(f)) != '\n'
       && c != EOF ; ++i) {
        if (i == (len - 2)) {
            *buff = realloc(*buff, (len += len/2) *
               sizeof(char));
        }
        *(*buff + i) = c;
    }
    *(*buff + i) = '\0';
    return i;
}

static int parseline(char *line, matrix_t **m1,
   matrix_t **m2) {
    char *linepos = line;
    int dim;
    int err;

    // get dimension
    if ((err = getInteger(&dim, &linepos)) != 0) {
        return err;
    }
    debug_print("dim:␣%d\n", dim);

    *m1 = create_matrix(dim, dim);
    if ((err = loadMatrix(&linepos, *m1)) != 0) {
        destroy_matrix(*m1);
        return err;
    }
    #ifdef DEBUG
    print_matrix_2d(*m1);
    #endif

    *m2 = create_matrix(dim, dim);
    if ((err = loadMatrix(&linepos, *m2)) != 0) {
        destroy_matrix(*m1);
        destroy_matrix(*m2);
        return err;
    }
    #ifdef DEBUG
    print_matrix_2d(*m2);
    #endif

    return 0;
}

static void advanceBlanks(char **pos) {
    while (**pos == ' ') {
        ++(*pos);
    }
}

static int invalidChar(char** pos) {
    return pos != NULL  && *pos != NULL && **pos != ' '
        && **pos != '\0';
}

static int getInteger(int *d, char **linepos) {
    advanceBlanks(linepos);
    errno = 0;
    long n = strtol(*linepos, linepos, 10);
    if (errno != 0 || invalidChar(linepos)) {
        return ERR_INVALID_INPUT;
    }
    *d = (int) n;
    return 0;
}

static int getDouble(double *d, char** linepos) {
    advanceBlanks(linepos);
    errno = 0;
    *d = strtod(*linepos, linepos);
    return (errno != 0 || invalidChar(linepos)) ?
       ERR_INVALID_INPUT : 0;
}

static int loadMatrix(char **linepos, matrix_t *m) {
    int i;
    for (i = 0; i < m->rows * m->cols; ++i) {
        if (getDouble(m->array + i, linepos) != 0) {
            return ERR_INVALID_INPUT;
        }
    }
    return 0;
}
```

engine.s

```
    .file         1 "src/engine.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .rdata
        .align   2
$LC0:
        .ascii   "parse error\n\000"
        .text
        .align   2
        .globl   run
        .ent     run
run:
        .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       $zero,24($fp)
        sw       $zero,28($fp)
        sw       $zero,32($fp)
$L18:
        lhu     $v0,__sF+12
        srl     $v0,$v0,5
        andi    $v0,$v0,0x1
        beq     $v0,$zero,$L20
        b       $L17
$L20:
        li      $a0,15                  # 0xf
        la      $t9,malloc
        jal     $ra,$t9
        sw      $v0,36($fp)
        addu    $v0,$fp,36
        la      $a0,__sF
        move    $a1,$v0
        li      $a2,15                  # 0xf
        la      $t9,fgetline
        jal     $ra,$t9
        blez    $v0,$L21
        addu    $v0,$fp,28
        lw      $a0,36($fp)
        addu    $a1,$fp,24
        move    $a2,$v0
        la      $t9,parseline
        jal     $ra,$t9
        beq     $v0,$zero,$L22
        lw      $a0,36($fp)
        la      $t9,free
        jal     $ra,$t9
        la      $a0,__sF+176
        la      $a1,$LC0
        la      $t9,fprintf
        jal     $ra,$t9
        li      $a0,1                   # 0x1
        la      $t9,exit
        jal     $ra,$t9
$L22:
        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
$L21:
        lw      $a0,36($fp)
        la      $t9,free
        jal     $ra,$t9
        b       $L18
$L17:
        move    $sp,$fp
        lw      $ra,48($sp)
        lw      $fp,44($sp)
        addu    $sp,$sp,56
        j       $ra
        .end    run
        .size   run, .-run
        .align  2
        .ent    fgetline
fgetline:
        .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)
        sw      $a2,56($fp)
        sw      $zero,24($fp)
$L24:
        lw      $v0,56($fp)
        addu    $v1,$v0,-1
        lw      $v0,24($fp)
        slt     $v0,$v0,$v1
        beq     $v0,$zero,$L25
        lw      $a0,48($fp)
        la      $t9,fgetc
        jal     $ra,$t9
        sb      $v0,28($fp)
        lbu     $v0,28($fp)
        sll     $v0,$v0,24
        sra     $v1,$v0,24
        li      $v0,10                  # 0xa
        beq     $v1,$v0,$L25
        lb      $v1,28($fp)
        li      $v0,-1                  # 0
            xffffffffffffffff
        bne     $v1,$v0,$L27
        b       $L25
$L27:
        lw      $v0,56($fp)
        addu    $v1,$v0,-2
        lw      $v0,24($fp)
        bne     $v0,$v1,$L29
        lw      $s0,52($fp)
        lw      $a0,52($fp)
        lw      $v1,56($fp)
        sra     $v0,$v1,31
        srl     $v0,$v0,31
        addu    $v0,$v1,$v0
        sra     $v1,$v0,1
        lw      $v0,56($fp)
        addu    $v0,$v0,$v1
        sw      $v0,56($fp)
        lw      $a0,0($a0)
        move    $a1,$v0
        la      $t9,realloc
        jal     $ra,$t9
        sw      $v0,0($s0)
$L29:
        lw      $v0,52($fp)
        lw      $v1,0($v0)
        lw      $v0,24($fp)
        addu    $v1,$v1,$v0
        lbu     $v0,28($fp)
        sb      $v0,0($v1)
        lw      $v0,24($fp)
        addu    $v0,$v0,1
        sw      $v0,24($fp)
        b       $L24
$L25:
        lw      $v0,52($fp)
        lw      $v1,0($v0)
        lw      $v0,24($fp)
        addu    $v0,$v1,$v0
        sb      $zero,0($v0)
        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    fgetline
        .size   fgetline, .-fgetline
        .align  2
        .ent    parseline
parseline:
        .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)
        lw      $v0,56($fp)
        sw      $v0,24($fp)
        addu    $v0,$fp,28
        move    $a0,$v0
        addu    $a1,$fp,24
        la      $t9,getInteger
        jal     $ra,$t9
        sw      $v0,32($fp)
        lw      $v0,32($fp)
        beq     $v0,$zero,$L31
        lw      $v0,32($fp)
        sw      $v0,36($fp)
        b       $L30
$L31:
        lw      $a0,28($fp)
        lw      $a1,28($fp)
        la      $t9,create_matrix
        jal     $ra,$t9
        move    $v1,$v0
        lw      $v0,60($fp)
        sw      $v1,0($v0)
        lw      $v0,60($fp)
        addu    $a0,$fp,24
        lw      $a1,0($v0)
        la      $t9,loadMatrix
        jal     $ra,$t9
        sw      $v0,32($fp)
        lw      $v0,32($fp)
        beq     $v0,$zero,$L32
        lw      $v0,60($fp)
        lw      $a0,0($v0)
        la      $t9,destroy_matrix
        jal     $ra,$t9
        lw      $v0,32($fp)
        sw      $v0,36($fp)
        b       $L30
$L32:
        lw      $a0,28($fp)
        lw      $a1,28($fp)
        la      $t9,create_matrix
        jal     $ra,$t9
        move    $v1,$v0
        lw      $v0,64($fp)
        sw      $v1,0($v0)
        lw      $v0,64($fp)
        addu    $a0,$fp,24
        lw      $a1,0($v0)
        la      $t9,loadMatrix
        jal     $ra,$t9
        sw      $v0,32($fp)
        lw      $v0,32($fp)
        beq     $v0,$zero,$L33
        lw      $v0,60($fp)
        lw      $a0,0($v0)
        la      $t9,destroy_matrix
        jal     $ra,$t9
        lw      $v0,64($fp)
        lw      $a0,0($v0)
        la      $t9,destroy_matrix
        jal     $ra,$t9
        lw      $v0,32($fp)
        sw      $v0,36($fp)
        b       $L30
$L33:
        sw      $zero,36($fp)
$L30:
        lw      $v0,36($fp)
        move    $sp,$fp
        lw      $ra,48($sp)
        lw      $fp,44($sp)
        addu    $sp,$sp,56
        j       $ra
        .end    parseline
        .size   parseline, .-parseline
        .align  2
        .ent    advanceBlanks
advanceBlanks:
        .frame  $fp,16,$ra              # vars= 0, regs
          = 2/0, args= 0, extra= 8
        .mask   0x50000000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,16
        .cprestore 0
        sw      $fp,12($sp)
        sw      $gp,8($sp)
        move    $fp,$sp
        sw      $a0,16($fp)
$L35:
        lw      $v0,16($fp)
        lw      $v0,0($v0)
        lb      $v1,0($v0)
        li      $v0,32                  # 0x20
        beq     $v1,$v0,$L37
        b       $L34
$L37:
        lw      $v1,16($fp)
        lw      $v0,16($fp)
        lw      $v0,0($v0)
        addu    $v0,$v0,1
        sw      $v0,0($v1)
        b       $L35
$L34:
        move    $sp,$fp
        lw      $fp,12($sp)
        addu    $sp,$sp,16
        j       $ra
        .end    advanceBlanks
        .size   advanceBlanks, .-advanceBlanks
        .align  2
        .ent    invalidChar
invalidChar:
        .frame  $fp,24,$ra              # vars= 8, regs
```

```
          = 2/0, args= 0, extra= 8
        .mask     0x50000000,-4
        .fmask    0x00000000,0
        .set      noreorder
        .cpload   $t9
        .set      reorder
        subu      $sp,$sp,24
        .cprestore 0
        sw        $fp,20($sp)
        sw        $gp,16($sp)
        move      $fp,$sp
        sw        $a0,24($fp)
        sw        $zero,8($fp)
        lw        $v0,24($fp)
        beq       $v0,$zero,$L39
        lw        $v0,24($fp)
        lw        $v0,0($v0)
        beq       $v0,$zero,$L39
        lw        $v0,24($fp)
        lw        $v0,0($v0)
        lb        $v1,0($v0)
        li        $v0,32                   # 0x20
        beq       $v1,$v0,$L39
        lw        $v0,24($fp)
        lw        $v0,0($v0)
        lb        $v0,0($v0)
        beq       $v0,$zero,$L39
        li        $v0,1                    # 0x1
        sw        $v0,8($fp)
$L39:
        lw        $v0,8($fp)
        move      $sp,$fp
        lw        $fp,20($sp)
        addu      $sp,$sp,24
        j         $ra
        .end      invalidChar
        .size     invalidChar, .-invalidChar
        .align    2
        .ent      getInteger
getInteger:
        .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        $a0,52($fp)
            la        $t9,advanceBlanks
            jal       $ra,$t9
            la        $t9,__errno
            jal       $ra,$t9
            sw        $zero,0($v0)
            lw        $v0,52($fp)
            lw        $a0,0($v0)
            lw        $a1,52($fp)
            li        $a2,10                    # 0xa
            la        $t9,strtol
            jal       $ra,$t9
            sw        $v0,24($fp)
            la        $t9,__errno
            jal       $ra,$t9
            lw        $v0,0($v0)
            bne       $v0,$zero,$L42
            lw        $a0,52($fp)
            la        $t9,invalidChar
            jal       $ra,$t9
            bne       $v0,$zero,$L42
            b         $L41
$L42:
            li        $v0,1                     # 0x1
            sw        $v0,28($fp)
            b         $L40
$L41:
            lw      $v1,48($fp)
            lw      $v0,24($fp)
            sw      $v0,0($v1)
            sw      $zero,28($fp)
    $L40:
            lw      $v0,28($fp)
            move    $sp,$fp
            lw      $ra,40($sp)
            lw      $fp,36($sp)
            addu    $sp,$sp,48
            j       $ra
            .end    getInteger
            .size   getInteger, .-getInteger
            .align  2
            .ent    getDouble
    getDouble:
            .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      $a0,52($fp)
            la      $t9,advanceBlanks
            jal     $ra,$t9
            la      $t9,__errno
            jal     $ra,$t9
            sw      $zero,0($v0)
            lw      $v0,52($fp)
            lw      $a0,0($v0)
            lw      $a1,52($fp)
            la      $t9,strtod
            jal     $ra,$t9
            lw      $v0,48($fp)
            s.d     $f0,0($v0)
            sw      $zero,24($fp)
            la      $t9,__errno
            jal     $ra,$t9
            lw      $v0,0($v0)
            bne     $v0,$zero,$L45
            lw      $a0,52($fp)
            la      $t9,invalidChar
            jal     $ra,$t9
            bne     $v0,$zero,$L45
            b       $L44
    $L45:
            li      $v0,1                   # 0x1
            sw      $v0,24($fp)
    $L44:
            lw      $v0,24($fp)
            move    $sp,$fp
            lw      $ra,40($sp)
            lw      $fp,36($sp)
            addu    $sp,$sp,48
            j       $ra
            .end    getDouble
            .size   getDouble, .-getDouble
            .align  2
            .ent    loadMatrix
    loadMatrix:
            .frame  $fp,48,$ra              # vars= 8, regs
               = 3/0, args= 16, extra= 8
            .mask   0xd0000000,-8
            .fmask  0x00000000,0
            .set    noreorder
            .cpload $t9
            .set    reorder
            subu    $sp,$sp,48
            .cprestore 16
            sw      $ra,40($sp)
            sw      $fp,36($sp)
            sw      $gp,32($sp)
            move    $fp,$sp
```

```
            sw          $a0 ,48( $fp )
            sw          $a1 ,52( $fp )
            sw          $zero ,24( $fp )
$L47 :
            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 , $L50
            b           $L48
$L50 :
            lw          $a0 ,52( $fp )
            lw          $v0 ,24( $fp )
            sll         $v1 , $v0 ,3
            lw          $v0 ,8( $a0 )
            addu        $v0 , $v1 , $v0
            move        $a0 , $v0
            lw          $a1 ,48( $fp )
            la          $t9 , getDouble
            jal         $ra , $t9
            beq         $v0 , $zero , $L49
            li          $v0 ,1                          # 0x1
            sw          $v0 ,28( $fp )
            b           $L46
$L49 :
            lw          $v0 ,24( $fp )
            addu        $v0 , $v0 ,1
            sw          $v0 ,24( $fp )
            b           $L47
$L48 :
            sw          $zero ,28( $fp )
$L46 :
            lw          $v0 ,28( $fp )
            move        $sp , $fp
            lw          $ra ,40( $sp )
            lw          $fp ,36( $sp )
            addu        $sp , $sp ,48
```

```
        j          $ra
        .end       loadMatrix
        .size      loadMatrix, .-loadMatrix
        .ident     "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)
           "
```

app.c

```
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include "engine.h"

#define VERSION 1.0
#define NO_ARGS 0

static void print_help();
static void print_version();

const char* short_options = "hV";

static struct option options[] = {
    { "help", NO_ARGS, NULL, 'h' },
    { "version", NO_ARGS, NULL, 'V' },
    { 0, 0 }
};

int main(int argc, char** argv) {
        int param = 0;
        int index = 0;
        if((param = getopt_long(argc, argv,
           short_options, options, &index)) != -1) {
                switch (param) {
                        case 'h':
                                print_help();
                                break;
                        case 'V':
                                print_version();
                                break;
                        default:
                                print_help();
```

```
                                        break;
                        }
            } else {
                run();
        }
        return EXIT_SUCCESS;
}

static void print_version() {
        printf("Version: %-1.1f\n", VERSION);
}

static void print_help() {
            printf("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");
}
```

app.s

```
        .file           1 "src/app.c"
            .section .mdebug.abi32
            .previous
            .abicalls
            .rdata
            .align  2
$LC0:
            .ascii  "hV\000"
            .globl  short_options
            .data
            .align  2
            .type   short_options, @object
```

```
         .size    short_options, 4
short_options:
         .word    $LC0
         .rdata
         .align   2
$LC1:
         .ascii   "help\000"
         .align   2
$LC2:
         .ascii   "version\000"
         .data
         .align   2
         .type    options, @object
         .size    options, 48
options:
         .word    $LC1
         .word    0
         .word    0
         .word    104
         .word    $LC2
         .word    0
         .word    0
         .word    86
         .word    0
         .word    0
         .space   8
         .text
         .align   2
         .globl   main
         .ent     main
main:
         .frame   $fp,64,$ra              # vars= 16,
            regs= 3/0, args= 24, extra= 8
         .mask    0xd0000000,-8
         .fmask   0x00000000,0
         .set     noreorder
         .cpload  $t9
         .set     reorder
         subu     $sp,$sp,64
         .cprestore 24
         sw       $ra,56($sp)
```

```
        sw      $fp,52($sp)
        sw      $gp,48($sp)
        move    $fp,$sp
        sw      $a0,64($fp)
        sw      $a1,68($fp)
        sw      $zero,32($fp)
        sw      $zero,36($fp)
        addu    $v0,$fp,36
        sw      $v0,16($sp)
        lw      $a0,64($fp)
        lw      $a1,68($fp)
        lw      $a2,short_options
        la      $a3,options
        la      $t9,getopt_long
        jal     $ra,$t9
        sw      $v0,32($fp)
        lw      $v1,32($fp)
        li      $v0,-1                  # 0
            xffffffffffffffff
        beq     $v1,$v0,$L18
        lw      $v0,32($fp)
        sw      $v0,40($fp)
        li      $v0,86                  # 0x56
        lw      $v1,40($fp)
        beq     $v1,$v0,$L21
        li      $v0,104                 # 0x68
        lw      $v1,40($fp)
        beq     $v1,$v0,$L20
        b       $L22
$L20:
        la      $t9,print_help
        jal     $ra,$t9
        b       $L24
$L21:
        la      $t9,print_version
        jal     $ra,$t9
        b       $L24
$L22:
        la      $t9,print_help
        jal     $ra,$t9
        b       $L24
```

```
$L18:
        la      $t9,run
        jal     $ra,$t9
$L24:
        move    $v0,$zero
        move    $sp,$fp
        lw      $ra,56($sp)
        lw      $fp,52($sp)
        addu    $sp,$sp,64
        j       $ra
        .end    main
        .size   main, .-main
        .rdata
        .align  2
$LC3:
        .ascii  "Version:␣%-1.1f\n\000"
        .align  3
$LC4:
        .word   0
        .word   1072693248
        .text
        .align  2
        .ent    print_version
print_version:
        .frame  $fp,40,$ra              # vars= 0, regs
          = 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        l.d     $f0,$LC4
        la      $a0,$LC3
        mfc1    $a2,$f0
        mfc1    $a3,$f1
        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_version
        .size   print_version, .-print_version
        .rdata
        .align  2
$LC5:
        .ascii  "Usage:\n"
        .ascii  "␣␣tp0␣-h\n"
        .ascii  "␣␣tp0␣-V\n"
        .ascii  "␣␣tp0␣<␣in_file␣>␣out_file\n"
        .ascii  "Options:\n"
        .ascii  "␣␣-V,␣--version␣␣␣␣Print␣version␣and␣
           quit.\n"
        .ascii  "␣␣-h,␣--help␣␣␣␣␣␣␣Print␣this␣
           information␣and␣quit.\n"
        .ascii  "Examples:\n"
        .ascii  "␣␣tp0␣<␣in.txt␣>␣out.txt\n"
        .ascii  "␣␣cat␣in.txt␣|␣tp0␣>␣out.txt\n\000"
        .text
        .align  2
        .ent    print_help
print_help:
        .frame  $fp,40,$ra              # vars= 0, regs
           = 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        la      $a0,$LC5
        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_help
        .size   print_help, .-print_help
        .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)
           "
```

debug.h

```
#ifdef DEBUG
#define debug_print(fmt, ...) fprintf(stderr, "[DEBUG] 
   "fmt, __VA_ARGS__)
#else
#define debug_print(fmt, ...)
#endif
```