




## Infraestructura básica

### Trabajo Práctico 0

Apellido y Nombre	Padrón	Correo electrónico
Blanco, Sebastian	98539	sebastian.e.blanco@gmail.com
Lavandeira, Lucas	98042	lucaslavandeira@gmail.com
Llauró, Manuel Luis	95736	llauromanuel@gmail.com

GitHub : <https://github.com/lucaslavandeira/palindrome>

# Índice

1. Introducción	2
2. Diseño e implementación	2
3. Modo de uso	2
4. Herramientas utilizadas y testing	3
5. Problemas encontrados	3
6. Correcciones realizadas	3
7. Casos de prueba	3
8. Archivo <code>run_tests.sh</code>	7
9. Anexo A: Código C	9
10. Anexo B: Código Assembly MIPS	15

## 1. Introducción

El objetivo de este trabajo práctico familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementando un programa para procesar archivos de texto por línea de comando: el programa recibirá los archivos o streams de entrada y salida, y deberá imprimir aquellas palabras del archivo de entrada (componentes léxicos) que sean palíndromos.

## 2. Diseño e implementación

Para lograr armar el programa se decidió utilizar el método en programación conocido como "divide y conquistarás", dividiendo el problema en una suma de problemas mas chicos que son resueltos por las distintas funciones creadas.

Primero se guardó en un array de char los caracteres que fueron pedidos en el enunciado que formen las palabras, considerando el resto de los caracteres separadores entre palabras. Y junto con este array se creó una función correspondiente la cual contesta si un caracter dado pertenece a una palabra o si es un separador de palabras.

Por otro lado se creó una función para saber si una palabra dada es o no un palíndromo, junto a otra función encargada de leer la palabra del archivo de entrada. Lo siguiente fue tratar el comportamiento esperado para el modo de uso del programa, como el modo de compilación y las entradas esperadas.

El programa fue desarrollado completamente en entornos GNU/Linux, bajo arquitecturas x86-64, teniendo en cuenta que el programa también debería poder ejecutarse en arquitecturas MIPS. Afortunadamente la simplicidad del programa, y de las herramientas de desarrollo del lenguaje de programación C, permitieron la transición entre arquitecturas sin problemas: el programa se compila y se comporta de la misma manera en ambos casos.

Con la suma de todo esto se logró armar el programa pedido.

## 3. Modo de uso

El ejecutable compilado no tiene dependencias con otros archivos o librerías, y puede moverse y ejecutarse desde cualquier directorio. Al ejecutarse desde una terminal sin argumentos adicionales, leerá de la entrada estándar palabras (es decir, componentes léxicos con caracteres alfanuméricos, y dígitos del 0 al 9), e imprimirá por la salida estándar aquellos que sean palíndromos. Al leer un carácter del final de archivo (EOF), finalizará su ejecución. El programa, adicionalmente, acepta varios parámetros adicionales (todos opcionales):

- **-h**: Muestra en pantalla los parámetros aceptados y finaliza su ejecución
- **-V**: Muestra la versión del programa compilado y finaliza su ejecución
- **-i <archivo>**: Lee la entrada del programa desde el archivo especificado
- **-o <archivo>**: Imprime la salida del programa al archivo especificado

El programa tiene dos códigos de salida: 0 en funcionamiento correcto, y 1 en caso de error, causado por la lectura inválida de un archivo de entrada, o escritura inválida del archivo de salida.

## 4. Herramientas utilizadas y testing

El funcionamiento correcto del proyecto se sometió a prueba haciendo uso de varias herramientas propias de los entornos Unix-like, principalmente de *bash*, y de las *coreutils* de GNU, para armar un simple script que busque archivos de entrada en un directorio, y compare los resultados (tanto la escritura de un archivo del parámetro `-o` como de la salida estándar) con archivos de salida. Para facilitar la compilación del programa (y del informe) se utilizó un simple Makefile. También se usa como compilador el designado por la cátedra, *gcc*.

## 5. Problemas encontrados

El desarrollo del programa no tuvo mayores inconvenientes, teniendo todos los integrantes experiencia programando en C. Como en todo proyecto, se debe explorar algunas tecnologías en las que uno mismo no está familiarizado, y se tuvo que dedicar un tiempo sustancial al estudio de *bash*, y *L<sup>A</sup>T<sub>E</sub>X*. La compilación del programa no es ejecutado con la mayor rigurosidad de advertencias debido a la detección de algunas cuando se compila en la plataforma MIPS. Sin embargo, el programa resulta funcionar de la misma manera: todos los casos de prueba pasan correctamente.

## 6. Correcciones realizadas

En esta segunda entrega del trabajo Práctico se soportan la entrada de palabras tan grandes como el archivo que se lee, o si proviene de la entrada estándar se soporta una palabra tan grande como la memoria lo permita. Además se definió, que dado el caso de si por alguna razón externa al programa, las funciones de la librería de C dan algún error, se finaliza el programa retornando el valor 1.

## 7. Casos de prueba

Se realizaron para el trabajo práctico 6 casos de pruebas distintos para verificar el correcto funcionamiento del código.

Para correr las pruebas se creó un archivo `run_tests.sh`, el cual corre todas las pruebas del directorio "test" (el mismo se lo puede encontrar el git, o en el pendrive del tp).

A continuación se presentan las pruebas realizadas:

Prueba: `"help.in"`

Argumentos de entrada: `-h`

La salida esperada en `"help.out"`: (vacío)

La salida esperada en `"help.stdout"`:

Usage:

`tp0 -h`

`tp0 -V`

`tp0 [options]`



La salida esperada en "long\_palindrome.out": El mismo archivo

La salida esperada en "long\_palindrome.stdout": (vacío)

Prueba: "long\_params.in"

Argumentos de entrada: --input test/single\_character.txt --output run.out

Archivo test/single\_character.txt:

A

La salida esperada en "long\_palindrome.out": A

La salida esperada en "long\_palindrome.stdout": (vacío)

Prueba: "long\_version.in"

Argumentos de entrada: --version

Archivo test/single\_character.txt:

La salida esperada en "long\_palindrome.out": (vacío)

La salida esperada en "long\_palindrome.stdout":

tp0: version 0.2

Prueba: "multiple\_palindrome.in"

Argumentos de entrada: -i test/multiple\_palindrome.txt -o run.out

Archivo test/multiple\_palindrome.txt:

Somos los primeros en completar el TP 0.

Ojo que La fecha de entrega del TP0 es el martes 12 de septiembre.

La salida esperada en "multiple\_palindrome.out":

Somos

0

Ojo

La salida esperada en "multiple\_palindrome.stdout": (vacío)

Prueba: "no\_palindrome.in"

Argumentos de entrada: -i test/no\_palindrome.txt -o run.out

Archivo test/no\_palindrome.txt:

Somos los primeros en completar el TP 0.

Ojo que La fecha de entrega del TP0 es el martes 12 de septiembre.

La salida esperada en "no\_palindrome.out":

Somos

0

Ojo

La salida esperada en "no\_palindrome.stdout": (vacío)

Prueba: "single\_character.in"

Argumentos de entrada: -i test/single\_character.txt -o run.out

Archivo test/single\_character.txt:

A

La salida esperada en "single\_character.out":

A

La salida esperada en "single\_character.stdout": (vacío)

Prueba: "underscores.in"

Argumentos de entrada: -i test/underscores.txt -o run.out

Archivo test/underscores.txt:

\_a\_ -a- a-b\_c\_b-a

La salida esperada en "underscores.out":

\_a\_

-a-

a-b\_c\_b-a

La salida esperada en "underscores.stdout": (vacío)

Prueba: "numeric.in"

Argumentos de entrada: -i test/numeric.txt -o run.out

Archivo test/numeric.txt:

1

9009

12

000000000000

La salida esperada en "numeric.out":

1

9009

000000000000

La salida esperada en "numeric.stdout": (vacío)

Prueba: "use\_stdout.in"

Argumentos de entrada: -i test/multiple\_palindrome.txt -o run.out

Archivo test/multiple\_palindrome.txt:

Somos los primeros en completar el TP 0.

Ojo que La fecha de entrega del TP0 es el martes 12 de septiembre.

La salida esperada en "numeric.out": (vacío)

La salida esperada en "numeric.stdout":

Somos

0

Ojo

Prueba: "version.in"

Argumentos de entrada: -V

La salida esperada en "numeric.out": (vacío)

La salida esperada en "numeric.stdout":

tp0: version 0.2

## 8. Archivo run\_tests.sh

---

```
#!/bin/bash

TEST_DIR=test/
IN=$TEST_DIR/in/
OUT=$TEST_DIR/out/
ERROR=$TEST_DIR/error/

for case in $(ls $ERROR); do
    rm $ERROR/$case
done

for case in $(ls $IN); do
    ERRORS=false
    if [ -e run.out ]; then
        rm run.out
    fi
    if [ -e stdout.tmp ]; then
        rm stdout.tmp
    fi
    filename=$(basename ${case%.*})
    printf "Executing $filename... ";
    ./tp0 $(cat $IN/$case) >> stdout.tmp
    if [ -e run.out ]; then
        diff run.out $OUT/$filename.out >> tmp;
        if [ $? -ne 0 ]; then
            ERRORS=true
        fi
    fi
fi
```



```
diff stdout.tmp $OUT/$filename.stdout >> tmp;
if [ $? -eq 0 ]; then
    echo "OK"
else
    ERRORS=true
fi

if $ERRORS; then
    echo "ERROR"
    cat tmp
    cp stdout.tmp $ERROR/$filename.error.stdout
    if [ -e run.out ]; then
        cp run.out $ERROR/$filename.error
    fi
fi
done

if [ -e tmp ]; then
    rm tmp
fi
if [ -e stdout.tmp ]; then
    rm stdout.tmp
fi
```

---

## 9. Anexo A: Código C

---

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdbool.h>
#include <stdlib.h>
#include <signal.h>
//-----
// DEFINITIONS
//-----
#define ERROR 1
#define SUCCESS 0
#define VERSION "0.2"
const char help_str[] = "Usage:\n"
    " tp0 -h\n"
    " tp0 -V\n"
    " tp0 [options]\n"
    "Options:\n"
    " -V, --version\tPrint version and quit.\n"
    " -h, --help\tPrint this information.\n"
    " -i, --input\tLocation of the input file.\n"
    " -o, --output\tLocation of the output file.\n"
    "Examples:\n"
    " tp0 -i ~/input -o\n";
#define SPACE_SIZE 65
#define SPACE_INDEX 123
#define EMPTY (-1)
const char ENTER = '\n';
char space[SPACE_SIZE];
int spaceIndex[SPACE_INDEX];
//-----
// CHARGE SPACE
//-----
// Del 97 al 122 estan las letras de a-z
// Del 65 al 90 estan las letras de A-Z
// Del 48 al 57 estan los numeros de 0-9
// '-' es 45
// '_' es 95
void chargeSpace() {
    int pos = 0;
    for (int i = 0; i < SPACE_INDEX; i++) spaceIndex[i] = EMPTY;
    //-----
    for (int i = 97; i <= 122; i++) {
        space[pos] = (char)i;
        spaceIndex[i] = pos;
        pos++;
    }
    //-----
    for (int i = 65; i <= 90; i++) {
        space[pos] = (char)i;
        spaceIndex[i] = pos;
        pos++;
    }
}
```

```

    }
    //-----
    for (int i = 48; i <= 57; i++) {
        space[pos] = (char)i;
        spaceIndex[i] = pos;
        pos++;
    }
    //-----
    // incluyo el guion medio
    pos++;
    space[pos] = '-';
    spaceIndex[45] = pos;
    //-----
    // incluyo el guion bajo
    pos++;
    space[pos] = '_';
    spaceIndex[95] = pos;
}
//-----
// BELONGS TO SPACE
//-----
bool belongsToSpace(int aChar) {
    if (aChar >= SPACE_INDEX) return false;
    return spaceIndex[aChar] != EMPTY;
}
//-----
// WRITE OUT FILE
//-----
int writeOutFile(FILE* archIn, long begin, long end, FILE* archOut) {
    int c;
    if (fseek(archIn, begin, SEEK_SET) < 0) return ERROR;
    for (long i = 0; i < (end - begin); i++) {
        if (fread((char*)&c, sizeof(char), 1, archIn) <= 0) return
            ERROR;
        if (fwrite((char*)&c, sizeof(char), 1, archOut) <= 0) return
            ERROR;
    }
    if (fwrite((char*)&ENTER, sizeof(char), 1, archOut) <= 0) return
        ERROR;
    return SUCCESS;
}
//-----
// READ LETTER
//-----
int readLetter(FILE* arch, long pos, char* letter) {
    if (fseek(arch, pos, SEEK_SET) == ERROR) return ERROR;
    if (fread(letter, sizeof(char), 1, arch) <= 0) return ERROR;
    return SUCCESS;
}
//-----
// CAPICUA
//-----
int capicua(FILE *archIn, long begin, long end, FILE *archOut) {
    char letterA, letterB;

```

```

bool isCapicua = true;
long size = (end - begin);
long leftPos = begin, rightPos = end-1;
if (size == 0) return SUCCESS;
while (leftPos <= rightPos) {
    if (readLetter(archIn, leftPos, &letterA) == ERROR) return ERROR;
    if (readLetter(archIn, rightPos, &letterB) == ERROR) return
        ERROR;
    int a = tolower(letterA);
    int b = tolower(letterB);
    if (a != b) {
        isCapicua = false;
        break;
    }
    rightPos--;
    leftPos++;
}
if (!isCapicua) return SUCCESS;
if (writeToFile(archIn, begin, end, archOut) == ERROR) return ERROR;
return SUCCESS;
}
//-----
// READ FILE
//-----
int readFile(FILE* archIn, FILE* archOut) {
    if (archIn != stdin) {
        if (fseek(archIn, 0, SEEK_END) == ERROR) return ERROR;
        if (ftell(archIn) == 0) return SUCCESS;
        if (fseek(archIn, 0, SEEK_SET) == ERROR) return ERROR;
    }

    if (archOut != stdout) {
        if (fseek(archOut, 0, SEEK_SET) == ERROR) return ERROR;
    }
    long begin = ftell(archIn);
    long end = begin;
    long last;
    int c = getc(archIn);
    bool first = false;
    while (c != EOF) {
        if (belongsToSpace(c)) {
            if (!first) {
                first = true;
                begin = ftell(archIn) - 1;
            }
            end = ftell(archIn);
        } else {
            first = false;
            last = ftell(archIn);
            if (capicua(archIn, begin, end, archOut) == ERROR) return
                ERROR;
            begin = end;
            if (fseek(archIn, last, SEEK_SET) == ERROR) return ERROR;
        }
    }
}

```

```

        c = getc(archIn);
    }
    long actualPos = ftell(archIn);
    if (fseek(archIn, 0, SEEK_END) == ERROR) return ERROR;
    long finalPos = ftell(archIn);
    if (actualPos != finalPos) return ERROR;
    return SUCCESS;
}
//-----
// EQUAL
//-----
bool equal(const char* str1, const char* str2) {
    return strcmp(str1, str2) == 0;
}
//-----
// ARG PARSE
//-----
int argParse(int argc, char** argv, FILE** descriptors, int* clean_exit)
{
    int arg = 1;
    const int size = 8;
    const char* flags[] = {"-i", "-o", "-V", "-h", "--version", "--help",
                           "--input", "--output"};

    bool std;
    char* flag = "";
    bool isFlagNull;
    while (arg < argc) {
        isFlagNull = true;
        if (argv[arg][0] == '-') {
            for (int i = 0; i < size; i++) {
                if (strcmp(argv[arg], flags[i]) == 0) {
                    flag = argv[arg];
                    isFlagNull = false;
                    break;
                }
            }
        }

        if (equal(flag, "-h") || equal(flag, "--help")) {
            printf("%s\n", help_str);
            *clean_exit = 1;
            return SUCCESS;
        }
        if (equal(flag, "-V") || equal(flag, "--version")) {
            printf("tp0: version %s\n", VERSION);
            *clean_exit = 1;
            return SUCCESS;
        }
        if (isFlagNull) {
            printf("Invalid argument: %s\n", argv[arg]);
            descriptors[0] = NULL;
            return ERROR;
        }
    } else {

```

```

        std = equal(argv[arg], "-");
        if ((equal(flag, "-i") || equal(flag, "--input")) && !std) {
            descriptors[0] = fopen(argv[arg], "r");
            if (descriptors[0] == NULL) return ERROR;
        } else if ((equal(flag, "-o") || equal(flag, "--output")) &&
            !std) {
            descriptors[1] = fopen(argv[arg], "w");
            if (descriptors[1] == NULL) return ERROR;
        }
        flag = "nullStr";
    }
    arg++;
}
return SUCCESS;
}

//-----
// STDIN CAPICUA
//-----
int stdinCapicua(const char* word, size_t len) {
    size_t cur = 0;
    while (cur < len) {
        if (tolower(word[cur]) != tolower(word[len - cur - 1])) {
            return false;
        }
        cur++;
    }
    return true;
}

//-----
// READ STDIN
//-----
// Funcin aparte para stdin, la optimizacin de memoria de fseek usada
// en el caso anterior no es posible leyendo de stdin
int readStdin(FILE* out) {
    size_t word_len = 1024;
    char* word = (char*) malloc(sizeof(char) * word_len);
    memset(word, 0, word_len);
    if (word == NULL) return ERROR;
    char c = (char) fgetc(stdin);
    size_t cur = 0;
    while (c != EOF) {
        if (belongsToSpace(c)) {
            if (cur >= word_len) {
                word_len *= 2;
                printf("ACA\n");
                word = (char*) realloc(word, word_len);
            }
            word[cur++] = c;
        } else {
            if (stdinCapicua(word, cur)) {
                if (fprintf(out, "%s\n", word) < 0) {
                    free(word);
                    return ERROR;
                }
            }
        }
    }
}

```

```

        }
        memset(word, 0, word_len);
        cur = 0;
    }
    c = (char) fgetc(stdin);
}
free(word);
return SUCCESS;
}
//-----
// MAIN
//-----
int main(int argc, char** argv) {
    FILE* fdescriptors[2] = {stdin, stdout};
    int clean_exit = 0;
    if (argParse(argc, argv, fdescriptors, &clean_exit) == ERROR) return
        1;
    if (clean_exit) return 0; // finalizacion limpia, cuando se usa -h o
        -v
    chargeSpace();
    if (fdescriptors[0] == stdin) {
        if (readStdin(fdescriptors[1]) == ERROR) return 1;
    } else {
        if (readFile(fdescriptors[0], fdescriptors[1]) == ERROR) return
            1;
        if (fclose(fdescriptors[0]) == EOF) return 1;
    }
    if (fdescriptors[1] != stdout && fclose(fdescriptors[1]) == EOF)
        return 1;
    return 0;
}
//-----

```

---

## 10. Anexo B: Código Assembly MIPS

Este código fue generado compilando el programa en la plataforma MIPS con los siguientes parámetros:

```
gcc -std=c99 -o0 -mrnames
```

Los parámetros `o0` y `mrnames` hacen que el código producido no tenga optimizaciones algunas, y reemplaza los nombres de los registros numéricos por los nombres convencionales usados en manuales de la arquitectura, respectivamente.

---

```
.file 1 "tp0.c"
.section .mdebug.abi32
.previous
.abicalls
.globl help_str
.rdata
.align 2
.type help_str, @object
.size help_str, 244
help_str:
.ascii "Usage:\n"
.ascii "  tp0 -h\n"
.ascii "  tp0 -V\n"
.ascii "  tp0 [options]\n"
.ascii "Options:\n"
.ascii "  -V, --version\tPrint version and quit.\n"
.ascii "  -h, --help\tPrint this information.\n"
.ascii "  -i, --input\tLocation of the input file.\n"
.ascii "  -o, --output\tLocation of the output file.\n"
.ascii "Examples:\n"
.ascii "  tp0 -i ~/input -o\n\000"
.globl ENTER
.type ENTER, @object
.size ENTER, 1
ENTER:
.byte 10
.text
.align 2
.globl chargeSpace
.ent chargeSpace
chargeSpace:
.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 $zero,8($fp)
sw $zero,12($fp)
$L18:
```



```

        lw $v0,12($fp)
        slt $v0,$v0,123
        bne $v0,$zero,$L21
        b $L19
$L21:
        lw $v0,12($fp)
        sll $v1,$v0,2
        la $v0,spaceIndex
        addu $v1,$v1,$v0
        li $v0,-1          # 0xffffffffffffffff
        sw $v0,0($v1)
        lw $v0,12($fp)
        addu $v0,$v0,1
        sw $v0,12($fp)
        b $L18
$L19:
        li $v0,97          # 0x61
        sw $v0,12($fp)
$L22:
        lw $v0,12($fp)
        slt $v0,$v0,123
        bne $v0,$zero,$L25
        b $L23
$L25:
        lw $v1,8($fp)
        la $v0,space
        addu $v1,$v1,$v0
        lbu $v0,12($fp)
        sb $v0,0($v1)
        lw $v0,12($fp)
        sll $v1,$v0,2
        la $v0,spaceIndex
        addu $v1,$v1,$v0
        lw $v0,8($fp)
        sw $v0,0($v1)
        lw $v0,8($fp)
        addu $v0,$v0,1
        sw $v0,8($fp)
        lw $v0,12($fp)
        addu $v0,$v0,1
        sw $v0,12($fp)
        b $L22
$L23:
        li $v0,65          # 0x41
        sw $v0,12($fp)
$L26:
        lw $v0,12($fp)
        slt $v0,$v0,91
        bne $v0,$zero,$L29
        b $L27
$L29:
        lw $v1,8($fp)
        la $v0,space
        addu $v1,$v1,$v0

```

```

    lbu $v0,12($fp)
    sb $v0,0($v1)
    lw $v0,12($fp)
    sll $v1,$v0,2
    la $v0,spaceIndex
    addu $v1,$v1,$v0
    lw $v0,8($fp)
    sw $v0,0($v1)
    lw $v0,8($fp)
    addu $v0,$v0,1
    sw $v0,8($fp)
    lw $v0,12($fp)
    addu $v0,$v0,1
    sw $v0,12($fp)
    b $L26
$L27:
    li $v0,48      # 0x30
    sw $v0,12($fp)
$L30:
    lw $v0,12($fp)
    slt $v0,$v0,58
    bne $v0,$zero,$L33
    b $L31
$L33:
    lw $v1,8($fp)
    la $v0,space
    addu $v1,$v1,$v0
    lbu $v0,12($fp)
    sb $v0,0($v1)
    lw $v0,12($fp)
    sll $v1,$v0,2
    la $v0,spaceIndex
    addu $v1,$v1,$v0
    lw $v0,8($fp)
    sw $v0,0($v1)
    lw $v0,8($fp)
    addu $v0,$v0,1
    sw $v0,8($fp)
    lw $v0,12($fp)
    addu $v0,$v0,1
    sw $v0,12($fp)
    b $L30
$L31:
    lw $v0,8($fp)
    addu $v0,$v0,1
    sw $v0,8($fp)
    lw $v1,8($fp)
    la $v0,space
    addu $v1,$v1,$v0
    li $v0,45      # 0x2d
    sb $v0,0($v1)
    lw $v0,8($fp)
    sw $v0,spaceIndex+180
    lw $v0,8($fp)

```

```

    addu $v0,$v0,1
    sw $v0,8($fp)
    lw $v1,8($fp)
    la $v0,space
    addu $v1,$v1,$v0
    li $v0,95      # 0x5f
    sb $v0,0($v1)
    lw $v0,8($fp)
    sw $v0,spaceIndex+380
    move $sp,$fp
    lw $fp,20($sp)
    addu $sp,$sp,24
    j $ra
.end chargeSpace
.size chargeSpace, .-chargeSpace
.align 2
.globl belongsToSpace
.ent belongsToSpace
belongsToSpace:
    .frame $fp,24,$ra    # vars= 8, regs= 2/0, args= 0, extra= 8
    .mask 0x50000000,-4
    .fmask 0x00000000,0
    .set noreorder
    .cplod $t9
    .set reorder
    subu $sp,$sp,24
    .cpstore 0
    sw $fp,20($sp)
    sw $gp,16($sp)
    move $fp,$sp
    sw $a0,24($fp)
    lw $v0,24($fp)
    slt $v0,$v0,123
    bne $v0,$zero,$L35
    sw $zero,8($fp)
    b $L34
$L35:
    lw $v0,24($fp)
    sll $v1,$v0,2
    la $v0,spaceIndex
    addu $v0,$v1,$v0
    lw $v1,0($v0)
    li $v0,-1      # 0xffffffffffffffff
    xor $v0,$v1,$v0
    sltu $v0,$zero,$v0
    sw $v0,8($fp)
$L34:
    lw $v0,8($fp)
    move $sp,$fp
    lw $fp,20($sp)
    addu $sp,$sp,24
    j $ra
.end belongsToSpace
.size belongsToSpace, .-belongsToSpace

```

```

.align 2
.globl writeOutFile
.ent writeOutFile
writeOutFile:
    .frame $fp,56,$ra    # vars= 16, regs= 3/0, args= 16, extra= 8
    .mask 0xd0000000,-8
    .fmask 0x00000000,0
    .set noreorder
    .cpld $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)
    sw $a3,68($fp)
    lw $a0,56($fp)
    lw $a1,60($fp)
    move $a2,$zero
    la $t9,fseek
    jal $ra,$t9
    bgez $v0,$L37
    li $v0,1          # 0x1
    sw $v0,32($fp)
    b $L36
$L37:
    sw $zero,28($fp)
$L38:
    lw $v1,64($fp)
    lw $v0,60($fp)
    subu $v1,$v1,$v0
    lw $v0,28($fp)
    slt $v0,$v0,$v1
    bne $v0,$zero,$L41
    b $L39
$L41:
    addu $a0,$fp,24
    li $a1,1          # 0x1
    li $a2,1          # 0x1
    lw $a3,56($fp)
    la $t9,fread
    jal $ra,$t9
    bne $v0,$zero,$L42
    li $v0,1          # 0x1
    sw $v0,32($fp)
    b $L36
$L42:
    addu $a0,$fp,24
    li $a1,1          # 0x1
    li $a2,1          # 0x1

```

```

        lw $a3,68($fp)
        la $t9,fwrite
        jal $ra,$t9
        bne $v0,$zero,$L40
        li $v0,1      # 0x1
        sw $v0,32($fp)
        b $L36
$L40:
        lw $v0,28($fp)
        addu $v0,$v0,1
        sw $v0,28($fp)
        b $L38
$L39:
        la $a0,ENTER
        li $a1,1      # 0x1
        li $a2,1      # 0x1
        lw $a3,68($fp)
        la $t9,fwrite
        jal $ra,$t9
        bne $v0,$zero,$L44
        li $v0,1      # 0x1
        sw $v0,32($fp)
        b $L36
$L44:
        sw $zero,32($fp)
$L36:
        lw $v0,32($fp)
        move $sp,$fp
        lw $ra,48($sp)
        lw $fp,44($sp)
        addu $sp,$sp,56
        j $ra
        .end writeOutFile
        .size writeOutFile, .-writeOutFile
        .align 2
        .globl readLetter
        .ent readLetter
readLetter:
        .frame $fp,48,$ra # vars= 8, regs= 3/0, args= 16, extra= 8
        .mask 0xd0000000,-8
        .fmask 0x00000000,0
        .set noreorder
        .cplload $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 $a2,56($fp)
        lw $a0,48($fp)

```

```

        lw $a1,52($fp)
        move $a2,$zero
        la $t9,fseek
        jal $ra,$t9
        move $v1,$v0
        li $v0,1      # 0x1
        bne $v1,$v0,$L46
        li $v0,1      # 0x1
        sw $v0,24($fp)
        b $L45
$L46:
        lw $a0,56($fp)
        li $a1,1      # 0x1
        li $a2,1      # 0x1
        lw $a3,48($fp)
        la $t9,fread
        jal $ra,$t9
        bne $v0,$zero,$L47
        li $v0,1      # 0x1
        sw $v0,24($fp)
        b $L45
$L47:
        sw $zero,24($fp)
$L45:
        lw $v0,24($fp)
        move $sp,$fp
        lw $ra,40($sp)
        lw $fp,36($sp)
        addu $sp,$sp,48
        j $ra
        .end readLetter
        .size readLetter, .-readLetter
        .align 2
        .globl capicua
        .ent capicua
capicua:
        .frame $fp,72,$ra    # vars= 32, regs= 3/0, args= 16, extra= 8
        .mask 0xd0000000,-8
        .fmask 0x00000000,0
        .set noreorder
        .cpld $t9
        .set reorder
        subu $sp,$sp,72
        .cprestore 16
        sw $ra,64($sp)
        sw $fp,60($sp)
        sw $gp,56($sp)
        move $fp,$sp
        sw $a0,72($fp)
        sw $a1,76($fp)
        sw $a2,80($fp)
        sw $a3,84($fp)
        li $v0,1      # 0x1
        sb $v0,26($fp)

```

```

    lw $v1,80($fp)
    lw $v0,76($fp)
    subu $v0,$v1,$v0
    sw $v0,28($fp)
    lw $v0,76($fp)
    sw $v0,32($fp)
    lw $v0,80($fp)
    addu $v0,$v0,-1
    sw $v0,36($fp)
    lw $v0,28($fp)
    bne $v0,$zero,$L50
    sw $zero,48($fp)
    b $L48
$L50:
    lw $v0,32($fp)
    lw $v1,36($fp)
    slt $v0,$v1,$v0
    beq $v0,$zero,$L52
    b $L51
$L52:
    lw $a0,72($fp)
    lw $a1,32($fp)
    addu $a2,$fp,24
    la $t9,readLetter
    jal $ra,$t9
    move $v1,$v0
    li $v0,1      # 0x1
    bne $v1,$v0,$L53
    li $v0,1      # 0x1
    sw $v0,48($fp)
    b $L48
$L53:
    addu $v0,$fp,25
    lw $a0,72($fp)
    lw $a1,36($fp)
    move $a2,$v0
    la $t9,readLetter
    jal $ra,$t9
    move $v1,$v0
    li $v0,1      # 0x1
    bne $v1,$v0,$L54
    li $v0,1      # 0x1
    sw $v0,48($fp)
    b $L48
$L54:
    lb $v0,24($fp)
    sll $v1,$v0,1
    lw $v0,_tolower_tab_
    addu $v0,$v1,$v0
    addu $v0,$v0,2
    lh $v0,0($v0)
    sw $v0,40($fp)
    lb $v0,25($fp)
    sll $v1,$v0,1

```

```

        lw $v0,_tolower_tab_
        addu $v0,$v1,$v0
        addu $v0,$v0,2
        lh $v0,0($v0)
        sw $v0,44($fp)
        lw $v1,40($fp)
        lw $v0,44($fp)
        beq $v1,$v0,$L55
        sb $zero,26($fp)
        b $L51
$L55:
        lw $v0,36($fp)
        addu $v0,$v0,-1
        sw $v0,36($fp)
        lw $v0,32($fp)
        addu $v0,$v0,1
        sw $v0,32($fp)
        b $L50
$L51:
        lbu $v0,26($fp)
        bne $v0,$zero,$L56
        sw $zero,48($fp)
        b $L48
$L56:
        lw $a0,72($fp)
        lw $a1,76($fp)
        lw $a2,80($fp)
        lw $a3,84($fp)
        la $t9,writeOutFile
        jal $ra,$t9
        move $v1,$v0
        li $v0,1      # 0x1
        bne $v1,$v0,$L57
        li $v0,1      # 0x1
        sw $v0,48($fp)
        b $L48
$L57:
        sw $zero,48($fp)
$L48:
        lw $v0,48($fp)
        move $sp,$fp
        lw $ra,64($sp)
        lw $fp,60($sp)
        addu $sp,$sp,72
        j $ra
        .end capicua
        .size capicua,.-capicua
        .align 2
        .globl readFile
        .ent readFile
readFile:
        .frame $fp,72,$ra    # vars= 32, regs= 3/0, args= 16, extra= 8
        .mask 0xd0000000,-8
        .fmask 0x00000000,0

```



```

.set noreorder
.cpload $t9
.set reorder
subu $sp,$sp,72
.cprestore 16
sw $ra,64($sp)
sw $fp,60($sp)
sw $gp,56($sp)
move $fp,$sp
sw $a0,72($fp)
sw $a1,76($fp)
lw $v1,72($fp)
la $v0,___sF
beq $v1,$v0,$L59
lw $a0,72($fp)
move $a1,$zero
li $a2,2      # 0x2
la $t9,fseek
jal $ra,$t9
move $v1,$v0
li $v0,1      # 0x1
bne $v1,$v0,$L60
li $v0,1      # 0x1
sw $v0,44($fp)
b $L58
$L60:
lw $a0,72($fp)
la $t9,ftell
jal $ra,$t9
bne $v0,$zero,$L61
sw $zero,44($fp)
b $L58
$L61:
lw $a0,72($fp)
move $a1,$zero
move $a2,$zero
la $t9,fseek
jal $ra,$t9
move $v1,$v0
li $v0,1      # 0x1
bne $v1,$v0,$L59
li $v0,1      # 0x1
sw $v0,44($fp)
b $L58
$L59:
lw $v1,76($fp)
la $v0,___sF+88
beq $v1,$v0,$L63
lw $a0,76($fp)
move $a1,$zero
move $a2,$zero
la $t9,fseek
jal $ra,$t9
move $v1,$v0

```

```

        li $v0,1      # 0x1
        bne $v1,$v0,$L63
        li $v0,1      # 0x1
        sw $v0,44($fp)
        b $L58
$L63:
        lw $a0,72($fp)
        la $t9,ftell
        jal $ra,$t9
        sw $v0,24($fp)
        lw $v0,24($fp)
        sw $v0,28($fp)
        lw $v1,72($fp)
        lw $v0,72($fp)
        lw $v0,4($v0)
        addu $v0,$v0,-1
        sw $v0,4($v1)
        bgez $v0,$L65
        lw $a0,72($fp)
        la $t9,___srget
        jal $ra,$t9
        sw $v0,48($fp)
        b $L66
$L65:
        lw $v0,72($fp)
        lw $v1,0($v0)
        move $a0,$v1
        lbu $a0,0($a0)
        sw $a0,48($fp)
        addu $v1,$v1,1
        sw $v1,0($v0)
$L66:
        lw $v0,48($fp)
        sw $v0,36($fp)
        sb $zero,40($fp)
$L67:
        lw $v1,36($fp)
        li $v0,-1      # 0xffffffffffffffff
        bne $v1,$v0,$L69
        b $L68
$L69:
        lw $a0,36($fp)
        la $t9,belongsToSpace
        jal $ra,$t9
        beq $v0,$zero,$L70
        lbu $v0,40($fp)
        bne $v0,$zero,$L71
        li $v0,1      # 0x1
        sb $v0,40($fp)
        lw $a0,72($fp)
        la $t9,ftell
        jal $ra,$t9
        addu $v0,$v0,-1
        sw $v0,24($fp)

```

```

$L71:
    lw $a0,72($fp)
    la $t9,ftell
    jal $ra,$t9
    sw $v0,28($fp)
    b $L72

$L70:
    sb $zero,40($fp)
    lw $a0,72($fp)
    la $t9,ftell
    jal $ra,$t9
    sw $v0,32($fp)
    lw $a0,72($fp)
    lw $a1,24($fp)
    lw $a2,28($fp)
    lw $a3,76($fp)
    la $t9,capicua
    jal $ra,$t9
    move $v1,$v0
    li $v0,1      # 0x1
    bne $v1,$v0,$L73
    li $v0,1      # 0x1
    sw $v0,44($fp)
    b $L58

$L73:
    lw $v0,28($fp)
    sw $v0,24($fp)
    lw $a0,72($fp)
    lw $a1,32($fp)
    move $a2,$zero
    la $t9,fseek
    jal $ra,$t9
    move $v1,$v0
    li $v0,1      # 0x1
    bne $v1,$v0,$L72
    li $v0,1      # 0x1
    sw $v0,44($fp)
    b $L58

$L72:
    lw $v1,72($fp)
    lw $v0,72($fp)
    lw $v0,4($v0)
    addu $v0,$v0,-1
    sw $v0,4($v1)
    bgez $v0,$L75
    lw $a0,72($fp)
    la $t9,___srget
    jal $ra,$t9
    sw $v0,52($fp)
    b $L76

$L75:
    lw $v0,72($fp)
    lw $v1,0($v0)
    move $a0,$v1

```

```

        lbu $a0,0($a0)
        sw $a0,52($fp)
        addu $v1,$v1,1
        sw $v1,0($v0)
$L76:
        lw $v0,52($fp)
        sw $v0,36($fp)
        b $L67
$L68:
        sw $zero,44($fp)
$L58:
        lw $v0,44($fp)
        move $sp,$fp
        lw $ra,64($sp)
        lw $fp,60($sp)
        addu $sp,$sp,72
        j $ra
        .end readFile
        .size readFile, .-readFile
        .align 2
        .globl equal
        .ent equal
equal:
        .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
        .cpstore 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 $a0,48($fp)
        lw $a1,52($fp)
        la $t9,strcmp
        jal $ra,$t9
        sw $v0,24($fp)
        lw $v1,24($fp)
        xori $v0,$v1,0x0
        sltu $v0,$v0,1
        sw $v0,24($fp)
        lw $v0,24($fp)
        move $sp,$fp
        lw $ra,40($sp)
        lw $fp,36($sp)
        addu $sp,$sp,48
        j $ra
        .end equal
        .size equal, .-equal

```

```

        .rdata
        .align 2
$LC0:
        .ascii  "-i\000"
        .align 2
$LC1:
        .ascii  "-o\000"
        .align 2
$LC2:
        .ascii  "-v\000"
        .align 2
$LC3:
        .ascii  "-h\000"
        .align 2
$LC4:
        .ascii  "--version\000"
        .align 2
$LC5:
        .ascii  "--help\000"
        .align 2
$LC6:
        .ascii  "--input\000"
        .align 2
$LC7:
        .ascii  "--output\000"
        .data
        .align 2
$LC8:
        .word $LC0
        .word $LC1
        .word $LC2
        .word $LC3
        .word $LC4
        .word $LC5
        .word $LC6
        .word $LC7
        .rdata
        .align 2
$LC9:
        .ascii  "\000"
        .align 2
$LC10:
        .ascii  "%s\n\000"
        .align 2
$LC11:
        .ascii  "tp0: version %s\n\000"
        .align 2
$LC12:
        .ascii  "0.2\000"
        .align 2
$LC13:
        .ascii  "Invalid argument: %s\n\000"
        .align 2
$LC14:

```

```

        .ascii  "-\000"
        .align  2
$LC15:
        .ascii  "r\000"
        .align  2
$LC16:
        .ascii  "w\000"
        .align  2
$LC17:
        .ascii  "nullStr\000"
        .text
        .align  2
        .globl  argParse
        .ent    argParse
argParse:
        .frame  $fp,104,$ra  # vars= 64, regs= 3/0, args= 16, extra= 8
        .mask  0xd0000000,-8
        .fmask 0x00000000,0
        .set   noreorder
        .cpld  $t9
        .set   reorder
        subu   $sp,$sp,104
        .cprestore 16
        sw     $ra,96($sp)
        sw     $fp,92($sp)
        sw     $gp,88($sp)
        move   $fp,$sp
        sw     $a0,104($fp)
        sw     $a1,108($fp)
        sw     $a2,112($fp)
        sw     $a3,116($fp)
        li     $v0,1      # 0x1
        sw     $v0,24($fp)
        li     $v0,8      # 0x8
        sw     $v0,28($fp)
        lw     $v0,$LC8
        sw     $v0,32($fp)
        lw     $v0,$LC8+4
        sw     $v0,36($fp)
        lw     $v0,$LC8+8
        sw     $v0,40($fp)
        lw     $v0,$LC8+12
        sw     $v0,44($fp)
        lw     $v0,$LC8+16
        sw     $v0,48($fp)
        lw     $v0,$LC8+20
        sw     $v0,52($fp)
        lw     $v0,$LC8+24
        sw     $v0,56($fp)
        lw     $v0,$LC8+28
        sw     $v0,60($fp)
        la     $v0,$LC9
        sw     $v0,68($fp)
$L79:

```

```

    lw $v0,24($fp)
    lw $v1,104($fp)
    slt $v0,$v0,$v1
    bne $v0,$zero,$L81
    b $L80
$L81:
    li $v0,1      # 0x1
    sb $v0,72($fp)
    lw $v0,24($fp)
    sll $v1,$v0,2
    lw $v0,108($fp)
    addu $v0,$v1,$v0
    lw $v0,0($v0)
    lb $v1,0($v0)
    li $v0,45     # 0x2d
    bne $v1,$v0,$L82
    sw $zero,76($fp)
$L83:
    lw $v0,76($fp)
    lw $v1,28($fp)
    slt $v0,$v0,$v1
    bne $v0,$zero,$L86
    b $L84
$L86:
    lw $v0,24($fp)
    sll $v1,$v0,2
    lw $v0,108($fp)
    addu $a0,$v1,$v0
    lw $v0,76($fp)
    sll $v1,$v0,2
    addu $v0,$fp,24
    addu $v0,$v1,$v0
    addu $v0,$v0,8
    lw $a0,0($a0)
    lw $a1,0($v0)
    la $t9,strcmp
    jal $ra,$t9
    bne $v0,$zero,$L85
    lw $v0,24($fp)
    sll $v1,$v0,2
    lw $v0,108($fp)
    addu $v0,$v1,$v0
    lw $v0,0($v0)
    sw $v0,68($fp)
    sb $zero,72($fp)
    b $L84
$L85:
    lw $v0,76($fp)
    addu $v0,$v0,1
    sw $v0,76($fp)
    b $L83
$L84:
    lw $a0,68($fp)
    la $a1,$LC3

```

```

    la $t9,equal
    jal $ra,$t9
    bne $v0,$zero,$L89
    lw $a0,68($fp)
    la $a1,$LC5
    la $t9,equal
    jal $ra,$t9
    bne $v0,$zero,$L89
    b $L88
$L89:
    la $a0,$LC10
    la $a1,help_str
    la $t9,printf
    jal $ra,$t9
    lw $v1,116($fp)
    li $v0,1      # 0x1
    sw $v0,0($v1)
    sw $zero,80($fp)
    b $L78
$L88:
    lw $a0,68($fp)
    la $a1,$LC2
    la $t9,equal
    jal $ra,$t9
    bne $v0,$zero,$L91
    lw $a0,68($fp)
    la $a1,$LC4
    la $t9,equal
    jal $ra,$t9
    bne $v0,$zero,$L91
    b $L90
$L91:
    la $a0,$LC11
    la $a1,$LC12
    la $t9,printf
    jal $ra,$t9
    lw $v1,116($fp)
    li $v0,1      # 0x1
    sw $v0,0($v1)
    sw $zero,80($fp)
    b $L78
$L90:
    lbu $v0,72($fp)
    beq $v0,$zero,$L93
    lw $v0,24($fp)
    sll $v1,$v0,2
    lw $v0,108($fp)
    addu $v0,$v1,$v0
    la $a0,$LC13
    lw $a1,0($v0)
    la $t9,printf
    jal $ra,$t9
    lw $v0,112($fp)
    sw $zero,0($v0)

```



```

        li $v0,1      # 0x1
        sw $v0,80($fp)
        b $L78
$L82:
        lw $v0,24($fp)
        sll $v1,$v0,2
        lw $v0,108($fp)
        addu $v0,$v1,$v0
        lw $a0,0($v0)
        la $a1,$LC14
        la $t9,equal
        jal $ra,$t9
        sb $v0,64($fp)
        lw $a0,68($fp)
        la $a1,$LC0
        la $t9,equal
        jal $ra,$t9
        bne $v0,$zero,$L95
        lw $a0,68($fp)
        la $a1,$LC6
        la $t9,equal
        jal $ra,$t9
        bne $v0,$zero,$L95
        b $L94
$L95:
        lbu $v0,64($fp)
        bne $v0,$zero,$L94
        lw $v0,24($fp)
        sll $v1,$v0,2
        lw $v0,108($fp)
        addu $v0,$v1,$v0
        lw $a0,0($v0)
        la $a1,$LC15
        la $t9,fopen
        jal $ra,$t9
        move $v1,$v0
        lw $v0,112($fp)
        sw $v1,0($v0)
        lw $v0,112($fp)
        lw $v0,0($v0)
        bne $v0,$zero,$L97
        li $v0,1      # 0x1
        sw $v0,80($fp)
        b $L78
$L94:
        lw $a0,68($fp)
        la $a1,$LC1
        la $t9,equal
        jal $ra,$t9
        bne $v0,$zero,$L99
        lw $a0,68($fp)
        la $a1,$LC7
        la $t9,equal
        jal $ra,$t9

```

```

    bne $v0,$zero,$L99
    b $L97
$L99:
    lbu $v0,64($fp)
    bne $v0,$zero,$L97
    lw $v0,24($fp)
    sll $v1,$v0,2
    lw $v0,108($fp)
    addu $v0,$v1,$v0
    lw $a0,0($v0)
    la $a1,$LC16
    la $t9,fopen
    jal $ra,$t9
    move $v1,$v0
    lw $v0,112($fp)
    addu $v0,$v0,4
    sw $v1,0($v0)
    lw $v0,112($fp)
    addu $v0,$v0,4
    lw $v0,0($v0)
    bne $v0,$zero,$L97
    li $v0,1      # 0x1
    sw $v0,80($fp)
    b $L78
$L97:
    la $v0,$LC17
    sw $v0,68($fp)
$L93:
    lw $v0,24($fp)
    addu $v0,$v0,1
    sw $v0,24($fp)
    b $L79
$L80:
    sw $zero,80($fp)
$L78:
    lw $v0,80($fp)
    move $sp,$fp
    lw $ra,96($sp)
    lw $fp,92($sp)
    addu $sp,$sp,104
    j $ra
    .end argParse
    .size argParse,.-argParse
    .align 2
    .globl stdin_capicua
    .ent stdin_capicua
stdin_capicua:
    .frame $fp,24,$ra    # vars= 8, regs= 2/0, args= 0, extra= 8
    .mask 0x50000000,-4
    .fmask 0x00000000,0
    .set noreorder
    .cpld $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 $a1,28($fp)
        sw $zero,8($fp)
$L102:
        lw $v0,8($fp)
        lw $v1,28($fp)
        sltu $v0,$v0,$v1
        bne $v0,$zero,$L104
        b $L103
$L104:
        lw $v1,24($fp)
        lw $v0,8($fp)
        addu $v0,$v1,$v0
        lb $v0,0($v0)
        sll $v1,$v0,1
        lw $v0,_tolower_tab_
        addu $v0,$v1,$v0
        addu $a0,$v0,2
        lw $v1,28($fp)
        lw $v0,8($fp)
        subu $v1,$v1,$v0
        lw $v0,24($fp)
        addu $v0,$v1,$v0
        addu $v0,$v0,-1
        lb $v0,0($v0)
        sll $v1,$v0,1
        lw $v0,_tolower_tab_
        addu $v0,$v1,$v0
        addu $v0,$v0,2
        lh $v1,0($a0)
        lh $v0,0($v0)
        beq $v1,$v0,$L105
        sw $zero,12($fp)
        b $L101
$L105:
        lw $v0,8($fp)
        addu $v0,$v0,1
        sw $v0,8($fp)
        b $L102
$L103:
        li $v0,1      # 0x1
        sw $v0,12($fp)
$L101:
        lw $v0,12($fp)
        move $sp,$fp
        lw $fp,20($sp)
        addu $sp,$sp,24
        j $ra
        .end stdin_capicua
        .size stdin_capicua, .-stdin_capicua

```

```

.align 2
.globl read_stdin
.ent read_stdin
read_stdin:
.frame $fp,64,$ra # vars= 24, regs= 3/0, args= 16, extra= 8
.mask 0xd0000000,-8
.fmask 0x00000000,0
.set noreorder
.cpload $t9
.set reorder
subu $sp,$sp,64
.cprestore 16
sw $ra,56($sp)
sw $fp,52($sp)
sw $gp,48($sp)
move $fp,$sp
sw $a0,64($fp)
li $v0,1024 # 0x400
sw $v0,24($fp)
lw $a0,24($fp)
la $t9,malloc
jal $ra,$t9
sw $v0,28($fp)
la $a0,__sF
la $t9,fgetc
jal $ra,$t9
sb $v0,32($fp)
sw $zero,36($fp)
$L107:
lb $v1,32($fp)
li $v0,-1 # 0xffffffffffffffff
bne $v1,$v0,$L109
b $L108
$L109:
lb $v0,32($fp)
move $a0,$v0
la $t9,belongsToSpace
jal $ra,$t9
beq $v0,$zero,$L110
lw $v1,36($fp)
lw $v0,24($fp)
bne $v1,$v0,$L111
lw $v0,24($fp)
sll $v0,$v0,1
sw $v0,24($fp)
lw $a0,28($fp)
lw $a1,24($fp)
la $t9,realloc
jal $ra,$t9
sw $v0,28($fp)
$L111:
addu $a1,$fp,36
lw $v1,0($a1)
move $a0,$v1

```

```

        lw $v0,28($fp)
        addu $a0,$a0,$v0
        lbu $v0,32($fp)
        sb $v0,0($a0)
        addu $v1,$v1,1
        sw $v1,0($a1)
        b $L112
$L110:
        lw $a0,28($fp)
        lw $a1,36($fp)
        la $t9,stdin_capicua
        jal $ra,$t9
        beq $v0,$zero,$L113
        lw $a0,64($fp)
        la $a1,$LC10
        lw $a2,28($fp)
        la $t9,fprintf
        jal $ra,$t9
        bgez $v0,$L113
        li $v0,1      # 0x1
        sw $v0,40($fp)
        b $L106
$L113:
        lw $a0,28($fp)
        move $a1,$zero
        lw $a2,24($fp)
        la $t9,memset
        jal $ra,$t9
        sw $zero,36($fp)
$L112:
        la $a0,__sF
        la $t9,fgetc
        jal $ra,$t9
        sb $v0,32($fp)
        b $L107
$L108:
        sw $zero,40($fp)
$L106:
        lw $v0,40($fp)
        move $sp,$fp
        lw $ra,56($sp)
        lw $fp,52($sp)
        addu $sp,$sp,64
        j $ra
        .end read_stdin
        .size read_stdin,.-read_stdin
        .data
        .align 2
$LC18:
        .word __sF
        .word __sF+88
        .text
        .align 2
        .globl main

```

```

.ent main
main:
.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 $v0,$LC18
sw $v0,24($fp)
lw $v0,$LC18+4
sw $v0,28($fp)
sw $zero,32($fp)
addu $v0,$fp,32
lw $a0,56($fp)
lw $a1,60($fp)
addu $a2,$fp,24
move $a3,$v0
la $t9,argParse
jal $ra,$t9
move $v1,$v0
li $v0,1          # 0x1
bne $v1,$v0,$L116
li $v0,1          # 0x1
sw $v0,36($fp)
b $L115
$L116:
lw $v0,32($fp)
beq $v0,$zero,$L117
sw $zero,36($fp)
b $L115
$L117:
la $t9,chargeSpace
jal $ra,$t9
lw $v1,24($fp)
la $v0, __sF
bne $v1,$v0,$L118
lw $a0,28($fp)
la $t9,read_stdin
jal $ra,$t9
beq $v0,$zero,$L121
lw $v1,28($fp)
la $v0, __sF+88
beq $v1,$v0,$L120
lw $a0,28($fp)
la $t9,fclose

```

```

        jal  $ra,$t9
$L120:
        li  $v0,1      # 0x1
        sw  $v0,36($fp)
        b   $L115
$L118:
        lw  $a0,24($fp)
        lw  $a1,28($fp)
        la  $t9,readFile
        jal  $ra,$t9
        move $v1,$v0
        li  $v0,1      # 0x1
        bne $v1,$v0,$L122
        li  $v0,1      # 0x1
        sw  $v0,36($fp)
        b   $L115
$L122:
        lw  $v1,24($fp)
        la  $v0, __sF
        beq  $v1,$v0,$L123
        lw  $a0,24($fp)
        la  $t9,fclose
        jal  $ra,$t9
        move $v1,$v0
        li  $v0,-1      # 0xffffffffffffffff
        bne $v1,$v0,$L123
        li  $v0,1      # 0x1
        sw  $v0,36($fp)
        b   $L115
$L123:
        lw  $v1,28($fp)
        la  $v0, __sF+88
        beq  $v1,$v0,$L121
        lw  $a0,28($fp)
        la  $t9,fclose
        jal  $ra,$t9
        move $v1,$v0
        li  $v0,-1      # 0xffffffffffffffff
        bne $v1,$v0,$L121
        li  $v0,1      # 0x1
        sw  $v0,36($fp)
        b   $L115
$L121:
        sw  $zero,36($fp)
$L115:
        lw  $v0,36($fp)
        move $sp,$fp
        lw  $ra,48($sp)
        lw  $fp,44($sp)
        addu $sp,$sp,56
        j   $ra
        .end main
        .size main, .-main

```

```
.comm space,65
```

```
.comm spaceIndex,492
```

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

---