

ИДЗ-3 Кокорев Артём  
Вариант 9

Файлы расположены в публичном репозитории: <https://github.com/w1sq/ABC>

Задание:

Разработать программу, которая «переворачивает на месте» заданную ASCII-строку символов (не копируя строку в другой буфер).

Код на 10 баллов:

main.asm

```
.include "macros.asm"
.include "utils.asm"

.data
prompt_in: .string "Введите имя входного файла: "
prompt_out: .string "Введите имя выходного файла: "
console_prompt: .string "Вывести результат на консоль? (Y/N): "
buffer: .space 2 # Для ответа Y/N

.text
.globl main
main:
# Сохраняем ra
addi sp, sp, -4
sw ra, 0(sp)
# Получаем имена файлов
jal get_filenames
# Открываем файлы и обрабатываем данные
jal process_files
# Спрашиваем про вывод на консоль
print_str(console_prompt)
# Читаем ответ
read_str(buffer, 2)
# Проверяем ответ
lb t0, buffer
li t1, 'Y'
beq t0, t1, print_to_console
```

```
li t1, 'y'
beq t0, t1, print_to_console
j end_program
```

#### *print\_to\_console:*

```
# Открываем выходной файл для чтения
open_file(filename_out, 0) # 0 – режим чтения
bltz a0, end_program
mv s0, a0 # сохраняем дескриптор
# Выделяем буфер для чтения
addi sp, sp, -256
mv s1, sp # сохраняем адрес буфера
```

#### *read\_and\_print:*

```
# Читаем из файла
read_file(s0, s1, 255)
beqz a0, cleanup # если достигнут конец файла
# Выводим на консоль
mv a1, a0 # сохраняем количество прочитанных байт
mv a0, s1
li a7, 4
ecall
j read_and_print
```

#### *cleanup:*

```
# Закрываем файл
close_file(s0)
# Освобождаем буфер
addi sp, sp, 256
```

#### *end\_program:*

```
# Восстанавливаем ra и завершаем программу
lw ra, 0(sp)
addi sp, sp, 4
li a7, 10
ecall
```

```
# Подпрограмма получения имен файлов
```

```

get_filenames:
addi sp, sp, -4
sw ra, 0(sp)
# Запрос имени входного файла
print_str(prompt_in)
read_str(filename_in, 256)
# Удаление \n из входного имени
la a0, filename_in
jal remove_newline
# Запрос имени выходного файла
print_str(prompt_out)
read_str(filename_out, 256)
# Удаление \n из выходного имени
la a0, filename_out
jal remove_newline
lw ra, 0(sp)
addi sp, sp, 4
ret

```

test.asm

```

.include "macros.asm"
.include "utils.asm"

.data
test_files: .string
"tests/test1.txt\0tests/test2.txt\0tests/test3.txt\0"
test_count: .word 3
test_out_suffix: .string "_out.txt\0"
test_msg: .string "Тестирование файла: "
tests_dir: .string "tests/"
debug_msg1: .string "Входной файл: "
debug_msg2: .string "Выходной файл: "

.text
.globl main
main:

```

```

# Вызываем тестовую программу
jal test_program
# Завершаем программу
li a7, 10
ecall

.globl test_program
test_program:
addi sp, sp, -4
sw ra, 0(sp)
lw t0, test_count
la t1, test_files
test_loop:
beqz t0, test_done
# Выводим имя тестируемого файла
print_str(test_msg)
mv a0, t1
li a7, 4
ecall
print_str(newline)
# Копируем имя входного файла
la a0, filename_in
mv a1, t1
jal copy_string
# Отладочный вывод входного файла
print_str(debug_msg1)
print_str(filename_in)
print_str(newline)
# Создаем имя выходного файла
la a0, filename_out
mv a1, t1
jal create_output_name
# Отладочный вывод выходного файла
print_str(debug_msg2)
print_str(filename_out)
print_str(newline)
# Обрабатываем файлы

```

```
jal process_files
# Следующий тест
addi t0, t0, -1
find_next:
lb t2, (t1)
addi t1, t1, 1
bnez t2, find_next
j test_loop
```

```
test_done:
lw ra, 0(sp)
addi sp, sp, 4
ret
```

```
# Копирование строки
```

```
copy_string:
addi sp, sp, -8
sw t0, 4(sp)
sw t1, 0(sp)
copy_loop:
lb t0, (a1)
sb t0, (a0)
beqz t0, copy_done
addi a0, a0, 1
addi a1, a1, 1
j copy_loop
copy_done:
lw t0, 4(sp)
lw t1, 0(sp)
addi sp, sp, 8
ret
```

```
# Создание имени выходного файла (добавляем _out.txt)
```

```
create_output_name:
addi sp, sp, -16
sw ra, 12(sp)
sw t0, 8(sp)
```

```

sw t1, 4(sp)
sw t2, 0(sp)
# Копируем входной путь полностью
mv t0, a0 # destination
mv t1, a1 # source
base_name_loop:
lb t2, (t1)
beqz t2, add_suffix # если конец строки
li t3, '.'
beq t2, t3, add_suffix # если нашли точку
sb t2, (t0)
addi t0, t0, 1
addi t1, t1, 1
j base_name_loop
add_suffix:
# Добавляем _out.txt
la t1, test_out_suffix
suffix_loop:
lb t2, (t1)
sb t2, (t0)
beqz t2, create_name_done
addi t0, t0, 1
addi t1, t1, 1
j suffix_loop
create_name_done:
lw ra, 12(sp)
lw t0, 8(sp)
lw t1, 4(sp)
lw t2, 0(sp)
addi sp, sp, 16
ret

```

utils.asm

```

.globl process_files
.globl remove_newline
.globl reverse_file
.globl file_error

```

```

.globl filename_in
.globl filename_out

.data
error_msg: .string "Ошибка при работе с файлом\n"
error_input: .string "Ошибка: входной файл не существует: "
error_output: .string "Ошибка: не удалось создать выходной
файл: "
newline: .string "\n"
filename_in: .space 256 # Буфер для имени входного файла
filename_out: .space 256 # Буфер для имени выходного файла

.text
# Подпрограмма обработки файлов
process_files:
# Сохраняем используемые регистры
addi sp, sp, -24
sw ra, 20(sp)
sw t3, 16(sp) # для дескриптора входного файла
sw t4, 12(sp) # для размера файла
sw t5, 8(sp) # для дескриптора выходного файла
sw t0, 4(sp) # для текущей позиции
sw t1, 0(sp) # для сравнения Y/N
# Открытие входного файла
la a0, filename_in
li a1, 0 # открываем для чтения
li a7, 1024
ecall
bltz a0, input_error
mv t3, a0
# Получение размера файла
mv a0, t3
li a1, 0
li a2, 2 # SEEK_END
li a7, 62
ecall
mv t4, a0 # сохраняем размер

```

```

# Возвращаемся в начало файла
mv a0, t3
li a1, 0
li a2, 0 # SEEK_SET
li a7, 62
ecall

# Открытие выходного файла
la a0, filename_out
li a1, 1 # открываем для записи
li a7, 1024
ecall
bgez a0, file_opened # если файл открылся успешно
# Создаем файл, если его нет
la a0, filename_out
li a1, 1 # создаем для записи
li a2, 0x1FF # права доступа (777 в восьмеричной системе)
li a7, 1024 # системный вызов для создания файла
ecall
bltz a0, output_error
file_opened:
mv t5, a0
# Делаем реверс файла
mv a0, t3 # входной дескриптор
mv a1, t5 # выходной дескриптор
mv a2, t4 # размер файла
jal reverse_file
# Закрываем файлы
mv a0, t3
li a7, 57
ecall
mv a0, t5
li a7, 57
ecall
# Восстанавливаем регистры
lw ra, 20(sp)
lw t3, 16(sp)
lw t4, 12(sp)

```



```
lw t5, 8(sp)
lw t0, 4(sp)
lw t1, 0(sp)
addi sp, sp, 24
ret
```

*# Подпрограмма реверса файла*

*reverse\_file:*

*# a0 – Входной дескриптор*

*# a1 – Выходной дескриптор*

*# a2 – размер файла*

```
addi sp, sp, -16
```

```
sw ra, 12(sp)
```

```
sw t0, 8(sp) # для позиции
```

```
sw t1, 4(sp) # для входного дескриптора
```

```
sw t2, 0(sp) # для выходного дескриптора
```

```
mv t1, a0 # сохраняем входной дескриптор
```

```
mv t2, a1 # сохраняем выходной дескриптор
```

```
addi t0, a2, -1 # начинаем с последней позиции
```

*# Выделяем место для символа*

```
addi sp, sp, -1
```

*read\_loop:*

```
bltz t0, reverse_done
```

*# Устанавливаем позицию*

```
mv a0, t1
```

```
mv a1, t0
```

```
li a2, 0
```

```
li a7, 62
```

```
ecall
```

*# Читаем символ*

```
mv a0, t1
```

```
mv a1, sp
```

```
li a2, 1
```

```
li a7, 63
```

```
ecall
```

*# Записываем символ*

```
mv a0, t2
mv a1, sp
li a2, 1
li a7, 64
ecall
addi t0, t0, -1
j read_loop
```

*reverse\_done:*

*# Освобождаем место символа*

```
addi sp, sp, 1
```

*# Восстанавливаем регистры*

```
lw ra, 12(sp)
```

```
lw t0, 8(sp)
```

```
lw t1, 4(sp)
```

```
lw t2, 0(sp)
```

```
addi sp, sp, 16
```

```
ret
```

*# Функция для удаления символа новой строки из строки*

*remove\_newline:*

```
addi sp, sp, -16
```

```
sw ra, 12(sp)
```

```
sw t0, 8(sp)
```

```
sw t1, 4(sp)
```

```
sw t2, 0(sp)
```

```
mv t0, a0 # Получаем адрес строки из a0
```

*remove\_loop:*

```
lb t1, (t0) # Загружаем текущий символ
```

```
beqz t1, remove_done
```

```
li t2, 10 # ASCII код новой строки
```

```
beq t1, t2, replace_newline
```

```
addi t0, t0, 1
```

```
j remove_loop
```

*replace\_newline:*

```
sb zero, (t0) # Заменяем \n на \0
```

*remove\_done:*

```
lw ra, 12(sp)
lw t0, 8(sp)
lw t1, 4(sp)
lw t2, 0(sp)
addi sp, sp, 16
ret
```

#### *file\_error:*

```
la a0, error_msg
li a7, 4
ecall
li a7, 10
ecall
```

#### *input\_error:*

```
la a0, error_input
li a7, 4
ecall
```

*# Выводим имя файла, который не удалось открыть*

```
la a0, filename_in
li a7, 4
ecall
la a0, newline
li a7, 4
ecall
li a7, 10
ecall
```

#### *output\_error:*

```
la a0, error_output
li a7, 4
ecall
```

*# Выводим имя файла, который не удалось создать*

```
la a0, filename_out
li a7, 4
ecall
la a0, newline
```

```
li a7, 4
ecall
li a7, 10
ecall
```

macros.asm

```
.macro print_str(%str)
la a0, %str
li a7, 4
ecall
.end_macro
```

```
.macro read_str(%buffer, %length)
la a0, %buffer
li a1, %length
li a7, 8
ecall
.end_macro
```

```
.macro open_file(%filename, %mode)
la a0, %filename
li a1, %mode
li a7, 1024
ecall
.end_macro
```

```
.macro close_file(%descriptor)
mv a0, %descriptor
li a7, 57
ecall
.end_macro
```

```
.macro seek_file(%descriptor, %position, %whence)
mv a0, %descriptor
mv a1, %position
li a2, %whence
```

```
li a7, 62
ecall
.end_macro
```

```
.macro read_file(%descriptor, %buffer, %count)
mv a0, %descriptor
mv a1, %buffer
li a2, %count
li a7, 63
ecall
.end_macro
```

```
.macro write_file(%descriptor, %buffer, %count)
mv a0, %descriptor
mv a1, %buffer
li a2, %count
li a7, 64
ecall
.end_macro
```

Makefile

```
ASM = rars
```

```
FLAGS = ae1 me sm
```

```
all: main test
```

```
main:
```

```
$(ASM) $(FLAGS) src/main.asm
```

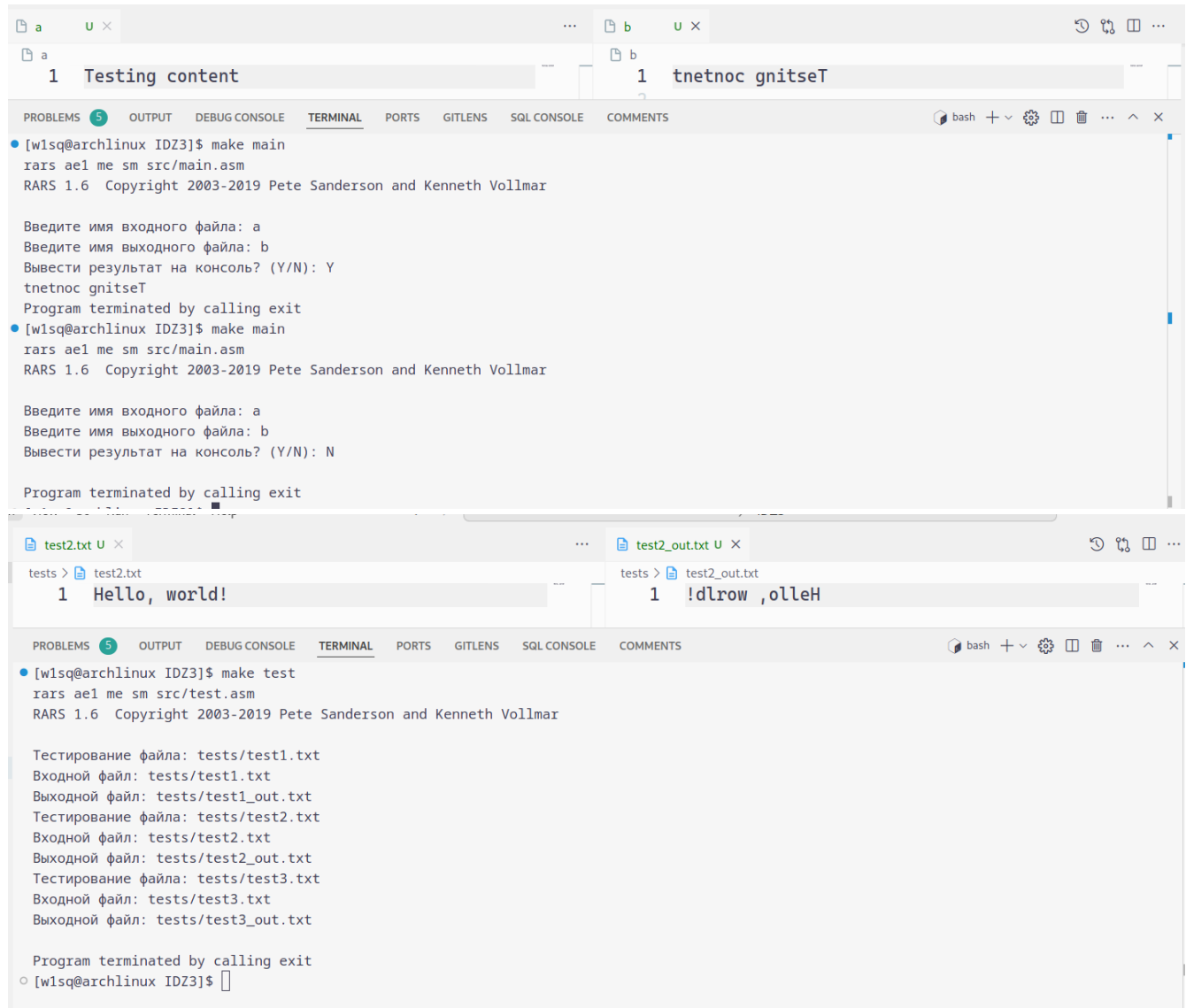
```
test:
```

```
$(ASM) $(FLAGS) src/test.asm
```

Принцип работы программы: идёт посимвольное считывание входного файла и запись в выходной файл. Ограничений на размеры рабочих файлов нет, т.к. работа с ними идёт посимвольно. Последний(3) тестовый файл весит 100 килобайт и отрабатывает за несколько секунд

Результаты запуска:

Программа состоит из основного блока с чтением и записью в считываемые из консоли файлы и тестирующего блока, который прогоняет программу по заранее заготовленным текстовым файлам из папки tests и генерирует \_out файлы соответственно.



```
[wlsq@archlinux IDZ3]$ make main
rais ae1 me sm src/main.asm
RARS 1.6 Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Введите имя входного файла: a
Введите имя выходного файла: b
Вывести результат на консоль? (Y/N): Y
tneT gnitnet
Program terminated by calling exit

[wlsq@archlinux IDZ3]$ make main
rais ae1 me sm src/main.asm
RARS 1.6 Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Введите имя входного файла: a
Введите имя выходного файла: b
Вывести результат на консоль? (Y/N): N

Program terminated by calling exit

tests > test2.txt
1 Hello, world!

tests > test2_out.txt
1 !dlrow ,olleH

[wlsq@archlinux IDZ3]$ make test
rais ae1 me sm src/test.asm
RARS 1.6 Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Тестирование файла: tests/test1.txt
Входной файл: tests/test1.txt
Выходной файл: tests/test1_out.txt
Тестирование файла: tests/test2.txt
Входной файл: tests/test2.txt
Выходной файл: tests/test2_out.txt
Тестирование файла: tests/test3.txt
Входной файл: tests/test3.txt
Выходной файл: tests/test3_out.txt

Program terminated by calling exit
[wlsq@archlinux IDZ3]$
```