Отчет ИДЗ_3

Отчет о проделанной работе

Выполнил Растворов Сергей БПИ 236

1. Описание проекта

В рамках данного проекта была реализована программа для обработки текстовых файлов. Программа читает содержимое входных файлов, обрабатывает данные (находит минимальные и максимальные ASCII значения символов), форматирует результат и записывает его в выходной файл. Также, добавлена возможность вывода результатов на консоль по запросу пользователя.

Процесс работы программы состоит из нескольких этапов:

1. Чтение входного файла.
2. Обработка текста (поиск минимального и максимального символов).
3. Форматирование строки с результатом.
4. Запись результата в выходной файл.
5. Вывод результата на консоль по запросу пользователя.

2. Реализованные функции и макросы

Основные части проекта были реализованы с использованием макросов, которые инкапсулируют часто используемые операции. Все макросы находятся в библиотеке `macros_lib.s`.

2.1 Чтение и запись файлов

- **read_file_name_macro**: Запрашивает у пользователя имя входного файла и проверяет корректность ввода.
- **read_file_macro**: Читает содержимое файла в кучу.
- **write_file_macro**: Записывает содержимое буфера в выходной файл.

2.2 Обработка данных

- **process_min_max_macro**: Обрабатывает строку, находя минимальный и максимальный ASCII символы.
- **format_srting_macro**: Форматирует строку с результатами обработки данных.

2.3 Взаимодействие с пользователем

- **print_result_console_dialog_macro**: Запрашивает у пользователя, хочет ли он вывести результаты на консоль. Если пользователь выбирает "Y", результаты выводятся на экран.

3. Автоматические тесты

Для автоматической проверки функциональности программы был реализован набор тестов. Каждый тест выполняет следующие шаги:

1. Чтение входного файла.
2. Обработка данных.
3. Запись результатов в выходной файл.

В процессе выполнения тестов, результаты автоматически сохраняются в выходные файлы в указанной директории.

В тестах используются различные входные данные, чтобы проверять корректность обработки возможных ситуаций.

Пример структуры теста:

- **input_file_name1**: Путь к входному файлу.
- **output_file_name1**: Путь к выходному файлу.
- **answer1**: Буфер для хранения результата обработки.

Для каждого теста вызывается макрос `run_test_case_macro`, который выполняет все необходимые операции.

Ручной / Автоматический запуск

Для успешного запуска ручного режима необходимо запустить `main.s`
в открывшееся поле для ввода, можете ввести путь до своего файла, если такого нет, можете воспользоваться базовым тестом `in.txt`

Для успешного запуска автотестов необходимо закомментировать `.globl main` и `main:` в файле `main.s` и запустить `auto_test.s`.

Ожидаемая оценка 10 – просроченный дедлайна = итоговая оценка за ИДЗ
:)

Код

`main.s`

```
.include "macros_lib.s"
```

```
 .eqv    NAME_SIZE 256  # Размер буфера для имени файла
 .eqv    TEXT_SIZE 512  # Размер буфера для текста
 .eqv    RESULT_SIZE 40

 .data
   input_file_name:      .space NAME_SIZE  # Имя читаемого файла
   output_file_name:     .space NAME_SIZE  # Имя читаемого файла
   buffer:               .space RESULT_SIZE        # Буфер для результата

 #
 # To run auto tests you need to comment
 # |.globl main| and |main:|
 # and run auto_test.s
 #



 .globl main
 .text
 main:
   read_file_name_macro("Please put input file name: ",input_file_name,
 NAME_SIZE, "Sorry, something wrong with filename")
   read_file_macro(input_file_name, TEXT_SIZE)
   process_min_max_macro a0
   format_srting_macro(a0, a1, buffer)
   read_file_name_macro("Please put output file name: ",output_file_name,
 NAME_SIZE, "Sorry, something wrong with filename")
   write_file_macro(output_file_name, buffer, RESULT_SIZE)
   print_result_console_dialog_macro("Do you want to print result on
 console ? Y/N", buffer, "Something wrong")
   end
```

`process_min_max.s`

```
 .include "macros_lib.s"
 .globl process_min_max
 .text
 process_min_max:
     # Входные параметры:
     # a0 - адрес начала строки
     # Выходные параметры:
     # a0 - код минимального символа
     # a1 - код максимального символа
     push(ra)
```

```
    mv t5, a0              # Сохраняем адрес начала строки
    li t1, 255             # максимальное значение для ASCII
    li t2, 0               # минимальное значение для ASCII

find_loop:
    lb t0, 0(t5)           # Считываем текущий символ строки
    beqz t0, end_p_m_m     # Если символ равен 0 (конец строки), завершаем
цикл

    # Проверка символа на допустимость
    li t3, 32              # Минимальный допустимый символ пробел
    blt t0, t3, find_next

    blt t0, t1, update_min # обновляем минимум
    bgt t0, t2, update_max # обновляем максимум
    j find_next

update_min:
    mv t1, t0
    j find_next

update_max:
    mv t2, t0
    j find_next

find_next:
    addi t5, t5, 1
    j find_loop

end_p_m_m:
    mv a0, t1
    mv a1, t2
    pop(ra)
    ret
```

`macros_lib.s`

```
.eqv    CHOICE_SIZE 3
.macro print_auto_answer_console_dialog(%message, %answer, %error)
.data
        dialog_message: .asciz %message
        choice_buffer:  .space CHOICE_SIZE
        y:              .asciz "Y"
```

```
.text
get_str_choice:
        la a0, dialog_message
        la a1, choice_buffer
        li a2, CHOICE_SIZE
        li a7, 54
        ecall

        bnez a1, not_correct_choice
        j correct_choice
not_correct_choice:
        message_dialog_macro(%error, 0)
        j get_str_choice
correct_choice:
        li      t4, '\n'
        la      t5, choice_buffer
choice_loop:
        lb      t6, (t5)
        beq     t4, t6, choice_replace
        addi    t5, t5, 1
        b       choice_loop
choice_replace:
        sb      zero, (t5)
        strcmp(y, choice_buffer)
        beqz    a0, print_answer
        j       choice_end
print_answer:
        la a0 %answer
        li a7 4
        ecall
choice_end:

.end_macro

################################################################################
################################################

.macro read_file_name_macro(%message, %file_name, %NAME_SIZE, %error)
.data
        message:                .asciz %message
.text
get_file_name:
        la a0, message
```

```
        la a1, %file_name
        li a2, %NAME_SIZE
        li a7, 54
        ecall

        li t1, -4
        beqz a1, correct_input_file_name
        beq a1, t1, not_correct_input_file_name
not_correct_input_file_name:
        message_dialog_macro(%error, 0)
        j get_file_name

correct_input_file_name:
        li      t4, '\n'
        la      t5, %file_name
read_input_file_name_loop:
        lb      t6, (t5)
        beq     t4, t6, read_input_file_name_replace
        addi    t5, t5, 1
        b       read_input_file_name_loop
read_input_file_name_replace:
        sb      zero, (t5)
        j       final_read_input_file_name
final_read_input_file_name:

.end_macro

################################################################################
################################################

.macro read_file_macro(%file_name, %TEXT_SIZE)
        la a0, %file_name
        li a1, %TEXT_SIZE
        jal read_file
.end_macro

################################################################################
################################################

.macro message_dialog_macro(%message, %type)
.data
        error_message: .asciz %message
.text
```

```
        la a0 error_message
        li a1 %type
        li a7 55
        ecall
.end_macro

################################################################################
################################################

## !! in s5 out s10-min s11-max !!
.macro process_min_max_macro %string_buffer
        mv      a0, %string_buffer
        jal process_min_max
.end_macro

################################################################################
#########################################################

.macro format_srting_macro(%first_symbol_code, %second_symbol_code,
%buffer)
.data
first_symbol:  .space 4
second_symbol:  .space 4

# in min min, max symbols and their ascii code
# <min symbol> - <ascii code>\n<max symbol> - <ascii code>
.text
    la s0, %buffer
    mv s1, %first_symbol_code
    mv s2, %second_symbol_code

    sb s1, (s0)
    addi s0, s0, 1

    store_symbol(' ')
    store_symbol('-')
    store_symbol(' ')

    char_to_ascii_code_string(s1, first_symbol)

    la s4, first_symbol
copy_loop_1:
    lb s3, (s4)
```

```
    beqz s3, end_copy_loop_1
    sb s3, (s0)
    addi s0, s0, 1
    addi s4, s4, 1
    j copy_loop_1
end_copy_loop_1:

    store_symbol('\n')

    sb s2, (s0)
    addi s0, s0, 1

    store_symbol(' ')
    store_symbol('-')
    store_symbol(' ')

    char_to_ascii_code_string(s2, second_symbol)

    la s4, second_symbol
copy_loop_2:
    lb s3, (s4)
    beqz s3, end_copy_loop_2
    sb s3, (s0)
    addi s0, s0, 1
    addi s4, s4, 1
    j copy_loop_2
end_copy_loop_2:

    sb zero, (s0)

.end_macro

################################################################################
#########################################################

# Записывает строку в буфер и добавляет нуль-терминатор в конец строки
# %strbuf - адрес буфера
# %size - размер строки, включая нуль-терминатор

.macro str_get(%strbuf, %size)
        la      a0, %strbuf
        li      a1, %size
        li      a7, 8
```

```
        ecall
        push(s0)
        push(s1)
        push(s2)
        li      s0, '\n'
        la      s1, %strbuf
next:
        lb      s2, (s1)
        beq     s0, s2, replace
        addi    s1, s1, 1
        b       next
replace:
        sb      zero, (s1)
        pop(s2)
        pop(s1)
        pop(s0)
.end_macro

#########################################################################
#########################################################

.macro read_addr_reg(%file_descriptor, %reg, %size)
        li      a7, 63
        mv      a0, %file_descriptor
        mv      a1, %reg
        mv      a2, %size
        ecall
.end_macro

#########################################################################
#########################################################

.macro write(%file_descriptor, %strbuf, %size)
        li      a7, 64                          # system call for write to
file
        mv      a0, %file_descriptor            # file descriptor
        mv      a1, %strbuf                     # address of buffer from
which to write
        mv      a2, %size                       # hardcoded buffer length
        ecall                                   # write to file
.end_macro

#########################################################################
```

```
#########################################################

.macro write_file_macro(%file_name, %buffer, %size)
        la      a0, %file_name
        la      a1, %buffer
        li      a2, %size
        jal     write_file
.end_macro

#########################################################################
#########################################################

.macro close(%file_descriptor)
        li      a7, 57
        mv      a0, %file_descriptor
        ecall
.end_macro

#########################################################################
#########################################################

.macro allocate(%size)
        li      a7, 9
        mv      a0, %size
        ecall
.end_macro

#########################################################################
#########################################################

.eqv READ_ONLY  0
.eqv WRITE_ONLY 1
.macro open(%file_name, %opt)
        li      a7, 1024
        mv      a0, %file_name
        li      a1, %opt
        ecall
        li      s1, -1
        beq     a0, s1, er_name
        j final_open
er_name:
        print_str("Incorrect file name\n")
        end
```

```
final_open:
.end_macro

################################################################################
#########################################################

.macro print_string %reg
        mv      a0, %reg
        li      a7, 4
        ecall
.end_macro

################################################################################
#########################################################

.macro print_str(%x)
        .data
str:
        .asciz %x
        .text
        push (a0)
        li a7, 4
        la a0, str
        ecall
        pop (a0)
.end_macro

################################################################################
#########################################################

.macro print_str_label %label
   la a0, %label
   li a7, 4
   ecall
.end_macro

################################################################################
#########################################################

.macro print_char %reg
        li      a7, 11
        mv      a0, %reg
        ecall
```

```
.end_macro

################################################################################
#########################################################

.macro print_int %reg
        mv      a0, %reg
        li      a7, 1
        ecall
.end_macro

################################################################################
#########################################################

.macro print_string_label %label
        la      a0, %label
        li      a7, 4
        ecall
.end_macro

################################################################################
#########################################################

# Сохраняет значение регистра в стек
.macro push(%x)
        addi    sp, sp, -4
        sw      %x, (sp)
.end_macro

################################################################################
#########################################################

# Восстанавливает значение из стека в регистр
.macro pop(%x)
        lw      %x, (sp)
        addi    sp, sp, 4
.end_macro

################################################################################
#########################################################

.macro end
        li      a7, 10
```

```
        ecall
.end_macro

#########################################################################
#########################################################

.macro write_ending_zero(%text_buffer, %size)
        # Добавляет нуль-терминатор в конец текстового буфера
        mv      t0, %text_buffer                # Адрес начала буфера
        add     t0, t0, %size                   # Смещаем на размер текста
        addi    t0, t0, 1                       # Переходим к следующему
байту
        sb      zero, (t0)                      # Записываем нуль-
терминатор в конец буфера
.end_macro

#########################################################################
#########################################################

.macro store_symbol(%symbol)
    li s3, %symbol
    sb s3, (s0)
    addi s0, s0, 1
.end_macro

#########################################################################
#########################################################

.macro char_to_ascii_code_string(%symbol_code, %buffer)
    # %symbol_code: код символа (для обработки)
    # %buffer: адрес выходной строки

    li t6, 0                    # t6 = длина (количество цифр)
    la t5, %buffer              # Загружаем начальный адрес буфера
    mv t4, %symbol_code         # Сохраняем код символа

    li t0, 10
convert_digit:
    beqz t4, reverse_digits
    rem t1, t4, t0          # t1 = t4 % 10
    addi t1, t1, '0'        # Конвертируем в ASCII
    sb t1, (t5)             # Сохраняем символ в буфер
    addi t5, t5, 1          # Смещаем указатель буфера
```

```
    addi t6, t6, 1        # Увеличиваем длину строки
    div t4, t4, t0        # t4 = t4 / 10
    j convert_digit

reverse_digits:
    sb zero, (t5)          # Добавляем нуль-терминатор
    addi t5, t5, -1        # Отступаем на один символ назад
    li t3, 2

    div t6, t6, t3
    la t0, %buffer         # t0 = начало строки
    li t4, 0

reverse_loop:
    beq t4, t6, reverse_done
    lb t2, (t0)
    lb t3, (t5)
    sb t3, (t0)
    sb t2, (t5)
    addi t0, t0, 1
    addi t5, t5, -1
    addi t4, t4, 1
    j reverse_loop

reverse_done:
.end_macro

##########################################################################
#########################################################

.macro run_test_case_macro(%input_file_name, %output_file_name, %answer)
.data
  error_filename: .asciz "Incorrect file name"
  error_bigfile: .asciz "Input file is too big"
  test_answer:  .space  256
.text
  open_for_test(%input_file_name, READ_ONLY)
  beqz  a0, error_filename_case
  read_file_macro(%input_file_name, 512)
  process_min_max_macro a0
  format_srting_macro(a0, a1, test_answer)
  #mv  s11, s0
  write_file_macro(%output_file_name, test_answer, 256)
```

```
 error_filename_case:
        strcmp(error_filename, %answer)
.end_macro

#########################################################################
#########################################################

.eqv  CHOICE_SIZE 3
.macro print_auto_answer_console_dialog(%message, %answer, %error)
.data
  dialog_message: .asciz %message
  choice_buffer:  .space CHOICE_SIZE
  y:    .asciz "Y"
.text
get_str_choice:
  la a0, dialog_message
  la a1, choice_buffer
  li a2, CHOICE_SIZE
  li a7, 54
  ecall

  bnez a1, not_correct_choice
  j correct_choice
not_correct_choice:
  message_dialog_macro(%error, 0)
  j get_str_choice
correct_choice:
      li  t4, '\n'
      la  t5, choice_buffer
choice_loop:
      lb  t6, (t5)
      beq   t4, t6, choice_replace
      addi   t5, t5, 1
      b  choice_loop
choice_replace:
      sb  zero, (t5)
      strcmp(y, choice_buffer)
      beqz  a0, print_answer
      j  choice_end
print_answer:
  la a0 %answer
  li a7 4
```

```
  ecall
choice_end:

.end_macro

##########################################################################
#########################################################

.eqv READ_ONLY  0  # Открыть для чтения
.eqv WRITE_ONLY  1  # Открыть для записи
.eqv APPEND  9  # Открыть для добавления
.macro open_for_test(%file_name, %opt)
      li     a7 1024           # Системный вызов открытия файла
      la      a0 %file_name       # Имя открываемого файла
      li     a1 %opt              # Открыть для чтения (флаг = 0)
      ecall                   # Дескриптор файла в a0 или -1)
      li  s1, -1        # Проверка на корректное открытие
      beq  a0, s1, er_name      # Ошибка открытия файла
      close(a0)
      li  a0, 1
      j final_open
er_name:
      li  a0, 0
final_open:
.end_macro

##########################################################################
#########################################################

.macro strcmp(%str1, %str2)
  la  a0, %str1
  la  a1, %str2
loop_strcmp:
      lb      t0 (a0)         # Загрузка символа из 1-й строки для
сравнения
      lb      t1 (a1)         # Загрузка символа из 2-й строки для
сравнения
      beqz    t0 end_strcmp        # Конец строки 1
      beqz    t1 end_strcmp        # Конец строки 2
      bne     t0 t1 end_strcmp     # Выход по неравенству
      addi    a0 a0 1         # Адрес символа в строке 1 увеличивается на
1
      addi    a1 a1 1         # Адрес символа в строке 2 увеличивается на
```

```
1
      j       loop_strcmp
end_strcmp:
      sub     a0 t0 t1        # Получение разности между символами
# Нв выход в регистре a0 ответ: 0 если равны, иначе 1
.end_macro

.eqv  CHOICE_SIZE 3
.macro print_result_console_dialog_macro(%message, %answer, %error)
.data
  dialog_message: .asciz %message
  choice_buffer:  .space CHOICE_SIZE
  y:    .asciz "Y"
.text
get_str_choice:
  la a0, dialog_message
  la a1, choice_buffer
  li a2, CHOICE_SIZE
  li a7, 54
  ecall

  bnez a1, not_correct_choice
  j correct_choice
not_correct_choice:
  message_dialog_macro(%error, 0)
  j get_str_choice
correct_choice:
      li  t4, '\n'
      la  t5, choice_buffer
choice_loop:
      lb  t6, (t5)
      beq   t4, t6, choice_replace
      addi   t5, t5, 1
      b  choice_loop
choice_replace:
      sb  zero, (t5)
      strcmp(y, choice_buffer)
      beqz  a0, print_answer
      j  choice_end
print_answer:
  la a0 %answer
  li a7 4
  ecall
```

```
choice_end:

.end_macro
```

auto_test.s

```
.include "macros_lib.s"
.eqv    TEXT_SIZE 512
.data
        input_file_name1:       .asciz "tests/input_text/test1.txt"
        input_file_name2:       .asciz "tests/input_text/test2.txt"
        input_file_name3:       .asciz "tests/input_text/test3.txt"
        input_file_name4:       .asciz "tests/input_text/test4.txt"
        input_file_name5:       .asciz "tests/input_text/test5.txt"
        input_file_name6:       .asciz "tests/input_text/test6.txt"

        output_file_name1:      .asciz "tests/output_text/test1.txt"
        output_file_name2:      .asciz "tests/output_text/test2.txt"
        output_file_name3:      .asciz "tests/output_text/test3.txt"
        output_file_name4:      .asciz "tests/output_text/test4.txt"
        output_file_name5:      .asciz "tests/output_text/test5.txt"
        output_file_name6:      .asciz "tests/output_text/test6.txt"

        answer1:        .space 256
        answer2:        .space 256
        answer3:        .space 256
        answer4:        .space 256
        answer5:        .space 256
        answer6:        .space 256

.globl auto_test
.text
auto_test:
        run_test_case_macro(input_file_name1, output_file_name1, answer1)
        run_test_case_macro(input_file_name2, output_file_name2, answer2)
        run_test_case_macro(input_file_name3, output_file_name3, answer3)
        run_test_case_macro(input_file_name4, output_file_name4, answer4)
        run_test_case_macro(input_file_name5, output_file_name5, answer5)
        run_test_case_macro(input_file_name6, output_file_name6, answer6)
        print_str("Auto tests done pls check dirictory
test/output_text\n")
```

```
        end
```

`read_file.s`

```
.include "macros_lib.s"

.globl read_file
.text
read_file:
         push(ra)
         push(s0)
         push(s1)
         push(s2)
         push(s3)
         push(s4)
         push(s5)
         push(s6)
         push(s7)
         push(s8)
         push(s9)
# В a0 лежит адрес буфера
# В a1 лежит размер текстовой части

        mv s6, a0 # В s6 помещаем адрес буфера
        mv s5, a1 # В s5 помещаем размер текстовой части

        open(s6, READ_ONLY)
        mv s0, a0 # В s0 помещаем дескриптор

 allocate(s5)
        mv s2, a0  # В s2 помещаем адрес кучи
        mv s3, a0  # В s3 помещаем текущий адрес кучи
        mv s4, s5  # В s4 помещаем размер текстовой части для контроля
выхода за пределы
        mv s7, zero # В s7 помещаем длину уже прочитанного текста, чтобы
добавить 0 в конце

        li s1, -1
        li s10, 1
        li s11, 20

read_loop:
```

```
        bgt s10, s11, end_read_loop # Проверяем превышение размера
        # Читаем данные из файла
        read_addr_reg(s0, s3, s5)
        beq a0, s1, end_read_loop # Если ошибка чтения
        mv    s8, a0         # Переносим размер прочитанного текста
        add  s7, s7, s8      # Увеличиваем длину прочитанного текста
        bne s8, s4, end_read_loop
        allocate(s5)
        add s3, s3, s4       # Перемещаем указатель для записи
        addi s10, s10, 1
        b read_loop

end_read_loop:
        close(s0)
        write_ending_zero(s2, s7)
        mv a0, s2

        pop(s9)
        pop(s8)
        pop(s7)
        pop(s6)
        pop(s5)
        pop(s4)
        pop(s3)
        pop(s2)
        pop(s1)
        pop(s0)
        pop(ra)
        ret
```

`write_file.s`

```
.include "macros_lib.s"
.global write_file
.text
write_file:
# В a0 лежит имя открываемого файла
# В a1 лежит адрес текста для записи
# В a2 лежит размер текста для записи

        push(ra)
        push(s3)
        push(s4)
```

```
        push(s5)
        push(s6)
        mv      s3, a0
        mv      s4, a1
        mv      s5, a2

        open(s3, WRITE_ONLY)
        mv      s6, a0          # save the file descriptor
        write(s6, s4, s5)

        close(s6)

        pop(s6)
        pop(s5)
        pop(s4)
        pop(s3)
        pop(ra)
        ret
```