

Зенин Вадим ИДЗ по АВС. Вариант -32

32. Разработать программу, которая на основе анализа двух входных ASCII-строк формирует на выходе две другие строки. В первой из выводимых строк содержатся символы, которых нет во второй исходной строке. Во второй выводимой строке содержатся символы, отсутствующие в первой входной строке (**разности символов**). Каждый символ в соответствующей выходной строке должен встречаться только один раз.

Данное ИДЗ было написано сразу на 10, поскольку семинарист разрешил.

Macrolib.asm

Библиотека макросов, которые использовались в ИДЗ.

```

2
3  # Печать строки передаваемой в макро.
4  .macro print_str(%x)
5      la a0, %x
6      li a7, 4
7      ecall
8  .end_macro
9
10 # Ввод строки с консоли
11 .macro read_str(%str, %bufsize)
12     la a0 %str
13     li a1 %bufsize
14     li a7 8
15     ecall
16 .end_macro
17
18
19 .macro read_str( %bufsize)
20     li a1 %bufsize
21     li a7 8
22     ecall
23 .end_macro
24
25 # Печать символа передаваемой в макро.
26 .macro print_char(%x)
27     push (a0)
28     li a7, 11
29     mv a0, %x
30     ecall
31     pop (a0)
32 .end_macro
33
34 # Перевод строки.
35 .macro newline
36 .data
37     newline: .asciz "\n"

```

```

37  newline: .ascii "\n"
38  .text
39          la a0, newline
40          li a7 4
41          ecall
42  .end_macro
43
44  # Завершение программы
45  .macro exit
46      li a7, 10
47      ecall
48  .end_macro
49
50  # Сохранение заданного регистра на стеке
51  .macro push(%x)
52      addi    sp, sp, -4
53      sw      %x, (sp)
54  .end_macro

```

```

56 # Выталкивание значения с вершины стека в регистр
57 .macro pop(%x)
58     lw     %x, (sp)
59     addi    sp, sp, 4
60 .end_macro
61
62 # Вывод поэлементно массива с адресом начала в "%x".
63 .macro print_array(%x, %size)
64     push(a2)
65     push(a3)
66     mv a3, zero
67
68     loop_print:
69         lb a2 (%x)
70         print_char(a2)
71         addi %x, %x, 4
72         addi a3, a3, 1
73         bgt %size, a3, loop_print
74         newline
75
76         pop(a3)
77         pop(a2)
78 .end_macro
79
80 .macro check_unique (%array, %x, %size)
81     # array - адрес массива
82     # x      - символ для проверки
83     # size   - количество элементов в массиве
84
85     push(t0)
86     push(t1)
87     push(t2)
88     li s11 1                # Предполагаем, что символ уникален
89     li t0, 0                # Счётчик итераций
90     la t1, %array           # Адрес массива
91
92     check_loop:
93         beq t0, %size, end_check # Если дошли до конца массива, завершаем
94         lb t2, 0(t1)             # Загружаем символ из массива
95         beq t2, %x, not_unique  # Если символ уже есть, он не уникален
96         addi t1, t1, 4           # Переходим к следующему элементу массива
97         addi t0, t0, 1           # Увеличиваем счётчик
98         j check_loop            # Повторяем цикл
99
100     not_unique:
101         li s11, 0              # Символ не уникален
102
103     end_check:
104         pop(t2)
105         pop(t1)
106         pop(t0)
107 .end_macro
108

```

```

110
111 #-----
112 # Ввод строки в буфер заданного размера с заменой перевода строки нулем
113 # %strbuf - адрес буфера
114 # %size - целая константа, ограничивающая размер вводимой строки
115 .macro str_get(%strbuf, %size)
116     la      a0 %strbuf
117     li      a1 %size
118     li      a7 8
119     ecall
120     push(s0)
121     push(s1)
122     push(s2)
123     li      s0 '\n'
124     la      s1      %strbuf
125 next:
126     lb      s2 (s1)
127     beq     s0      s2      replace
128     addi    s1 s1 1
129     b       next
130 replace:
131     sb      zero (s1)
132     pop(s2)
133     pop(s1)
134     pop(s0)
135 .end_macro
136
137
138 # Параметры
139 # a0 - строка для ввода имени файла.
140 #
141 # a0 - возврат дескриптор файла или -1.
142 #
143 #-----
144 # Открытие файла для чтения, записи, дополнения
145 .eqv READ_ONLY 0      # Открыть для чтения
146 .eqv WRITE_ONLY 1     # Открыть для записи
147 .eqv APPEND 9        # Открыть для добавления
148 .macro open(%file_name, %opt)
149     li      a7 1024      # Системный вызов открытия файла
150     la      a0 %file_name # Имя открываемого файла
151     li      a1 %opt      # Открыть для чтения (флаг = 0)
152     ecall      # Дескриптор файла в a0 или -1)
153 .end_macro
154
155 #-----

```

```

157 # Чтение информации из открытого файла
158 .macro read(%file_descriptor, %strbuf, %size)
159     li    a7, 63          # Системный вызов для чтения из файла
160     mv    a0, %file_descriptor # Дескриптор файла
161     la    a1, %strbuf      # Адрес буфера для читаемого текста
162     li    a2, %size        # Размер читаемой порции
163     ecall                # Чтение
164 .end_macro
165
166 -----
167 # Чтение информации из открытого файла,
168 # когда адрес буфера в регистре
169 .macro read_addr_reg(%file_descriptor, %reg, %size)
170     li    a7, 63          # Системный вызов для чтения из файла
171     mv    a0, %file_descriptor # Дескриптор файла
172     mv    a1, %reg        # Адрес буфера для читаемого текста из регистра
173     li    a2, %size        # Размер читаемой порции
174     ecall                # Чтение
175 .end_macro
176
177 -----
178 # Закрытие файла
179 .macro close(%file_descriptor)
180     li    a7, 57          # Системный вызов закрытия файла
181     mv    a0, %file_descriptor # Дескриптор файла
182     ecall                # Закрытие файла
183 .end_macro
184
185 -----
186 # Выделение области динамической памяти заданного размера
187 .macro allocate(%size)
188     li    a7, 9
189     li    a0, %size        # Размер блока памяти
190     ecall
191 .end_macro
192
193 # Данный макрос зануляет все элементы буфера по необходимым адресам и используется при реализации непрерывной работы программы.
194 .macro clear_buf(%obj, %size)
195     la    t0, %obj
196     li    t1, %size
197 clear_loop1:
198     beqz  t1, fin
199     sb    zero, (t0)
200     addi  t0, t0, 1
201     addi  t1, t1, -1
202     j     clear_loop1
203 fin:
204 .end_macro
205
206 -----
207
208 # Макрос для копирования строки
209 .macro strcpy (%s1, %s2)
210     la    a5 %s1
211     la    a6 %s2
212     jal   strcpy
213 .end_macro
214
215 # Данный макрос зануляет все элементы буфера по необходимым адресам и используется при реализации непрерывной работы программы.
216 .macro clear_buf(%obj, %size)
217     la    t0, %obj
218     li    t1, %size
219 clear_loop1:
220     beqz  t1, fin
221     sb    zero, (t0)
222     addi  t0, t0, 1
223     addi  t1, t1, -1
224     j     clear_loop1
225 fin:
226 .end_macro

```

```

206
207 # Макрос создает массив уникальных элементов из строки.
208 .macro create_array_uniq(%array,%str,%size)
209     push(s0)
210     push(t0)
211     push(t3)
212     mv     s0 %str
213     la     t0 %array
214 loop:
215
216     lb     t3 (s0)                # очередной символ
217     beqz   t3 fin                # нулевой – конец строки
218
219     check_unique(%array, t3, %size)    # Проверка элемента на уникальность
220     bnez s11, add_uniq
221     addi   s0 s0 1                # следующий символ
222     b      loop
223
224 add_uniq:
225     sw     t3, (t0)                # Загружаем значение a2 по адресу начала массива 1.
226     addi   t0, t0, 4                # Сдвигаем адрес в регистре t0 для следующего элемента.
227     addi   s0 s0 1                # следующий символ
228     addi   %size, %size 1          # Увеличиваю текущий размер массива уникальных элементов.
229     b      loop
230 fin:
231     pop(t3)
232     pop(t0)
233     pop(s0)
234
235
236 .end_macro
237
238
239 # Макрос, который создает строку разности элементов, сравнивая 2 уникальных массива между собой.
240
241 .macro compare(%newstr_buf, %array1,%size1, %array2, %size2)
242     push(s0)
243     push(t0)
244     push(t3)
245     push(t2)
246     push(t1)
247     la s0 %newstr_buf
248     la t0 %array1
249     mv t2 %size1
250     mv t1 zero
251 loop:
252
253     lb     t3 0(t0)                # очередной символ
254     addi   t0, t0, 4                # Сдвигаем адрес в регистре t0 для следующего элемента.
255     beq    t2, t1 fin                # нулевой – конец строки
256     addi   t1 t1 1
257     check_unique(%array2, t3, %size2)    # Проверка элемента на уникальность
258     bnez s11, update_newstr
259     b      loop
260
261 update_newstr:
262     sb     t3, (s0)                # Загружаем значение a2 по адресу начала массива 1.
263     addi   s0 s0 1                # следующий символ
264     b      loop
265 fin:
266     sb     zero, 0(s0)            # Завершаем строку символом 0
267     pop(t1)
268     pop(t2)
269     pop(t3)
270     pop(t0)
271     pop(s0)
272
273 .end_macro
274

```

Макросы используемые в программе:

1. print_str(%x) — выводит строку, адрес которой передан в %x.

2. `read_str(%str, %bufsize)` — ввод строки в буфер `%str` с ограничением размера `%bufsize`.
3. `print_char(%x)` — выводит символ (ASCII-код в `%x`).
4. `newline` — перевод строки (`\n`).
5. `exit` — завершает выполнение программы.
6. `push(%x)` — сохраняет значение регистра `%x` на стеке.
7. `pop(%x)` — извлекает значение из стека в регистр `%x`.
8. `check_unique(%array, %x, %size)` — проверяет, уникален ли элемент `%x` в массиве `%array`.
9. `str_get(%strbuf, %size)` — ввод строки в буфер с заменой `'\n'` на `'\0'`.
10. `open(%file_name, %opt)` — открывает файл `%file_name` с опцией `%opt`.
11. `read(%file_descriptor, %strbuf, %size)` — читает данные из файла в буфер `%strbuf`.
12. `close(%file_descriptor)` — закрывает файл.
13. `allocate(%size)` — выделяет динамическую память размера `%size`.
14. `strcpy(%s1, %s2)` — копирует строку `%s2` в `%s1`.
15. `clear_buf(%obj, %size)` — очищает буфер `%obj` размером `%size`.
16. `create_array_uniq(%array, %str, %size)` — создает массив уникальных элементов из строки `%str`.
17. `compare(%newstr_buf, %array1, %size1, %array2, %size2)` — создает строку разности между `%array1` и `%array2`.

Блок .data

```
2  .include "macrolib.asm"
3
4  .globl array_chars1, array_chars2, file_name newstr1 newstr2
5  .eqv    SIZE 512                # размер буфера
6
7  .data
8      file_name:    .space SIZE      # Имя читаемого файла
9      array_chars1: .space SIZE
10     array_chars2: .space SIZE
11     newstr1:      .space SIZE      # Буфер для новой строки, output
12     newstr2:      .space SIZE      # Буфер для новой строки, output
13     output_file_name1: .asciz "Введите имя для первого файла вывода строки (с расширением .txt): "
14     output_file_name2: .asciz "Введите имя для второго файла вывода строки (с расширением .txt): "
15     prompt:        .asciz "Введите путь до читаемого файла: "
16
17     choice_exit:    .asciz "Хотите ли вы завершить работу программы [Y/N]: "
18
```

Размер буфера = 512 байт

Файл main.asm

```

19 .text
20 j main          # При первом проходе скипаем, тк буфферы пустые.
21
22 #=====
23
24 # Чистим буфферы для повторного использования.
25 la t6 clear_buffers
26 jalr t6
27
28 main:
29 #=====
30
31 print_str(prompt)
32 la t6 read_name    # Ввод имени файла, который необходимо будет считать.
33 jalr t6
34 la t6 open_read_file
35 jalr t6
36 mv s8 a0           # Сохранение адреса начала буффера.
37 mv s7 a1           # Сохранение величины прочитанной строки.
38
39 print_str(prompt)
40 la t6 read_name    # Ввод имени файла, который необходимо будет считать.
41 jalr t6
42 la t6 open_read_file
43 jalr t6
44
45 mv s6 a0           # Сохранение адреса начала буффера.
46 mv s5 a1           # Сохранение величины прочитанной строки.
47
48 #=====
49
50 create_array_uniq(array_chars1, s8, s10) # Макрос для создания массива уникальных символов из строки
51 create_array_uniq(array_chars2, s6, s9)  # Макрос для создания массива уникальных символов из строки
52
53 compare(newstr1, array_chars1, s10, array_chars2, s9) # Макрос для создания строки с разностью символов строк путем сравнения 2 массивов.
54 compare(newstr2, array_chars2, s9, array_chars1, s10) # Макрос для создания строки с разностью символов строк путем сравнения 2 массивов.
55
56 #=====
57
58 la t6 choice
59 jalr t6
60
61 #=====
62
63
64 output:
65 --
66 output:
67 print_str(output_file_name1)
68 la t6 read_name    # Ввод имени файла, который необходимо будет сохранить.
69 jalr t6
70 la a1 newstr1      # Передаю параметры (a1 – адрес начала буффера строки, a2 – размер) в подпрограмму
71 mv a2 s10
72 la t6 save_file    # Подпрограмма для сохранения файлов
73 jalr t6
74
75 print_str(output_file_name2)
76 la t6 read_name    # Ввод имени файла, который необходимо будет сохранить.
77 jalr t6
78 la a1 newstr2      # Передаю параметры (a1 – адрес начала буффера строки, a2 – размер) в подпрограмму
79 mv a2 s9
80 la t6 save_file    # Подпрограмма для сохранения файлов
81 jalr t6
82
83 #=====
84
85 print_str(choice_exit) # Запрашиваю у пользователя хочет ли он продолжить работу программы
86 la t6 check_choice
87 jalr t6
88 beqz a0 clear_buffers # Если да, переходим в отдел чисты буфферов по новой.
89
90 exit
91
92 #=====

```

- Первым делом скипается очистка буфферов, поскольку при первом цикле программы это не нужно. При повторных циклах уже происходит очистка.
- Далее у пользователя запрашиваются имена 2 файлов, которые открываются и считываются после чего я перевожу адреса начала буфферов и их длину в соответствующие регистры, которые будут использоваться далее.

- Далее идет 1 из основных макросов create_array_uniq

```

207 # Макрос создает массив уникальных элементов из строки.
208 .macro create_array_uniq(%array,%str,%size)
209     push(s0)
210     push(t0)
211     push(t3)
212     mv     s0 %str
213     la     t0 %array
214 loop:
215
216     lb     t3 (s0)                # очередной символ
217     beqz   t3 fin                # нулевой – конец строки
218
219     check_unique(%array, t3, %size)    # Проверка элемента на уникальность
220     bnez s11, add_uniq
221     addi   s0 s0 1                # следующий символ
222     b      loop
223
224 add_uniq:
225     sw     t3, (t0)                # Загружаем значение a2 по адресу начала массива 1.
226     addi   t0, t0, 4                # Сдвигаем адрес в регистре t0 для следующего элемента.
227     addi   s0 s0 1                # следующий символ
228     addi   %size, %size 1          # Увеличиваю текущий размер массива уникальных элементов.
229     b      loop
230 fin:
231     pop(t3)
232     pop(t0)
233     pop(s0)
234
235
236 .end_macro

```

он создает массив уникальных элементов из строки. По следующему алгоритму: берем элемент из строки, если его еще нет в массив, то добавляем его, если есть, то скипаем. Таким образом мы получаем 2 уникальных массива из 1 и 2 строки.

- Далее идет макрос compare

```

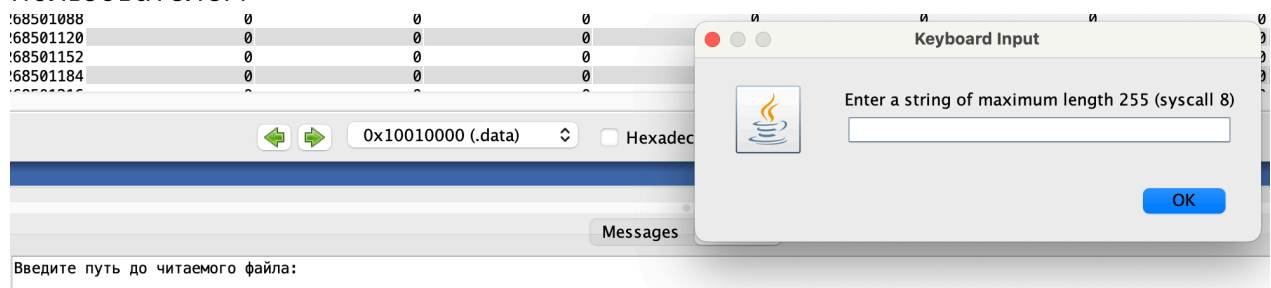
239 # Макрос, который создает строку разности элементов, сравнивая 2 уникальных массива между собой.
240
241 .macro compare(%newstr_buf, %array1,%size1, %array2, %size2)
242     push(s0)
243     push(t0)
244     push(t3)
245     push(t2)
246     push(t1)
247     la s0 %newstr_buf
248     la t0 %array1
249     mv t2 %size1
250     mv t1 zero
251 loop:
252
253     lb     t3 0(t0)                # очередной символ
254     addi   t0, t0, 4                # Сдвигаем адрес в регистре t0 для следующего элемента.
255     beq    t2, t1 fin                # нулевой – конец строки
256     addi   t1 t1 1
257     check_unique(%array2, t3, %size2)    # Проверка элемента на уникальность
258     bnez s11, update_newstr
259     b      loop
260
261 update_newstr:
262     sb     t3, (s0)                # Загружаем значение a2 по адресу начала массива 1.
263     addi   s0 s0 1                # следующий символ
264     b      loop
265 fin:
266     sb     zero, 0(s0)            # Завершаем строку символом 0
267     pop(t1)
268     pop(t2)
269     pop(t3)
270     pop(t0)
271     pop(s0)
272
273 .end_macro

```

Он сравнивает 2 массива уникальных элементов и создает строку разницы символов 1 и 2 массивов.

- Далее имея 2 строки с разностью символов запрашиваем у пользователя хочет ли он вывести результат на экран (да или нет). В зависимости от ответа выводим или нет.
- Далее запрашиваем 2 имя файлов (с расширением) у пользователя, в которые он хочет сохранить результат работы программы.
- После спрашиваем у пользователя хочет ли он завершить работу программы или нет. Если нет, то переходим к очистке буфферов, если да то завершаем программу.
- Программа также использует диалоговые окна при взаимодействии с

пользователем



Подпрограммы для обращения к файлам. Файл algorithms.asm

```

1  .include "macrolib.asm"
2  .globl read_name, open_read_file, save_file, choice, clear_buffers, check_choice, strcpy
3  .eqv NAME_SIZE 256 # Размер буфера для имени файла
4  .eqv TEXT_SIZE 512 # Размер буфера для текста
5  .data
6
7      er_name_mes: .asciz "Неверное имя файла\n"
8      er_read_mes: .asciz "Некорректная операция чтения файла\n"
9      er_choice_mes: .asciz "Некорректный ввод! Введите [Y/N]:"
10     choice_outres: .asciz "Хотите ли вы вывести результаты работы программы на консоль [Y/N]: "
11     output_nstr1: .asciz "Вот символы, которые есть в первой строке, но нет во второй: "
12     output_nstr2: .asciz "Вот символы, которые есть во второй строке, но нет в первой: "
13
14
15     solution: .space NAME_SIZE
16     Yes: .asciz "Y"
17     No: .asciz "N"
18
19 # Параметры
20 # a0 - строка для ввода имени файла.
21 #
22 # a0 - возврат начала буфера файла без \n на конце.
23 # a1 - возврат длины прочитанной строки.
24 #
25 open_read_file:
26     push(s0)
27     push(s1)
28     push(s2)
29     push(s3)
30     push(s4)
31     push(s5)
32     push(s6)
33     open(file_name, READ_ONLY)
34     li s1, -1 # Проверка на корректное открытие
35     beq a0, s1, er_name # Ошибка открытия файла
36     mv s0, a0 # Сохранение дескриптора файла
37
38     # Выделение начального блока памяти для буфера в куче
39     allocate(TEXT_SIZE) # Результат хранится в a0
40     mv s3, a0 # Сохранение адреса кучи в регистре
41     mv s5, a0 # Сохранение изменяемого адреса кучи в регистре
42     li s4, TEXT_SIZE # Сохранение константы для обработки
43     mv s6, zero # Установка начальной длины прочитанного текста
44
45 read_loop:
46     # Чтение информации из открытого файла
47     read_addr_reg(s0, s5, TEXT_SIZE) # чтение для адреса блока из регистра
48
49     # Проверка на корректное чтение
50     beq a0, s1, er_read # Ошибка
51     mv s2, a0 # Сохранение длины текста
52     add s6, s6, s2 # Размер текста увеличивается на прочитанную порцию
53
54     # Если длина считанного текста, чем размер буфера,
55     # нужно завершить процесс.
56     bne s2, s4, end_loop
57
58     # Иначе расширить буфер и повторить
59     allocate(TEXT_SIZE) # Результат здесь не нужен, но если нужно то...
60     add s5, s5, s2 # Адрес для чтения смещается на размер порции
61     b read_loop # Обработка следующей порции текста из файла
62
63 end_loop:
64     close(s0) # Закрытие файла
65     mv t0, s3 # Адрес буфера в куче
66     add t0, t0, s6 # Адрес последнего прочитанного символа
67     addi t0, t0, 1 # Место для нуля
68     sb zero, (t0) # Запись нуля в конец текста
69     mv a0, s3 # Переносу адрес буфера из s3 в a0
70     mv a1, s6 # Переносу размер прочитанной строки в a1
71
72     pop(s6)
73     pop(s5)

```

```

89     pop(s6)
90     pop(s5)
91     pop(s4)
92     pop(s3)
93     pop(s2)
94     pop(s1)
95     pop(s0)
96     ret
97
98
99
100  # Параметры
101  # a1 - Адрес буфера записываемого текста
102  # a2 - Размер записываемой порции из регистра
103  #
104  save_file:
105      push(s0)
106      push(s1)
107      push(s3)
108      push(s6)
109      mv s3 a1
110      mv s6 a2
111      # Сохранение прочитанного файла в другом файле
112      open(file_name, WRITE_ONLY)
113      li      s1      -1                # Проверка на корректное открытие
114      beq     a0      s1      er_name    # Ошибка открытия файла
115      mv      s0      a0                # Сохранение дескриптора файла
116
117      # Запись информации в открытый файл
118      li      a7,     64                # Системный вызов для записи в файл
119      mv      a0,     s0                # Дескриптор файла
120      mv      a1,     s3
121      mv      a2,     s6
122      ecall                                # Запись в файл
123
124      close(s0)
125      pop(s6)
126      pop(s3)
127      pop(s1)
128      pop(s0)
129
130      ret

```

```

135
136  er_name:
137      # Сообщение об ошибочном имени файла
138      la          a0      er_name_mes
139      li          a7      4
140      ecall
141      # И завершение программы
142      exit
143  er_read:
144      # Сообщение об ошибочном чтении
145      la          a0      er_read_mes
146      li          a7      4
147      ecall
148      # И завершение программы
149      exit
150
151  # Алгоритм для копирования строки.
152  strcpy:
153  loop_copy:
154      lb          t0, (a5)
155      sb          t0, (a6)
156      beqz        t0, end
157      addi        a5, a5, 1
158      addi        a6, a6, 1
159      b           loop_copy
160
161  end:
162      ret
163

```

```

167 # Параметры
168 #
169 # a0 - Результат 1 или 0.
170 #
171 check_choice:
172     push(s0)
173     push(s1)
174     la s0 Yes
175     la s1 No
176 loop_input_choice:
177     read_str(solution, TEXT_SIZE) # Считываю ответ пользователя
178     la a0 solution
179     lb t0 (a0)
180     lb t1 (s0)
181     lb t2 (s1)
182     beq t0 t1 yes # Сравниваю с Y и N если ответ не соответствует, то повторяю запрос ответа до корректного ввода
183     beq t0 t2 no
184     j incorrect_input
185
186
187
188 incorrect_input:
189     print_str(er_choice_mes)
190     j loop_input_choice
191 yes:
192     mv a0 zero # Если ответ да возвращаю 1 иначе 0
193     addi a0 a0 1
194     j fin
195 no:
196     mv a0 zero
197     j fin
198 fin:
199
200     pop(s1)
201     pop(s0)
202     ret
---
206 choice:
207     push(ra)
208     newline
209     print_str(choice_outres)
210     la t6 check_choice # Переход в подпрограмму для запросу выбора у пользователя.
211     jalr t6
212     beqz a0 finish
213
214     # Вывод результатов программы в консоль, если пользователь захочет.
215     newline
216     print_str(output_nstr1)
217     newline
218     print_str(newstr1) # Вывод первой строки результатов
219     newline
220     print_str(output_nstr2)
221     newline
222     print_str(newstr2) # Вывод второй строки результатов
223     newline
224     newline
225 finish:
226     pop(ra)
227     ret
228
229
230 # Подпрограмма для очистки буфферов.
231 clear_buffers:
232     clear_buf(array_chars1, TEXT_SIZE)
233     clear_buf(array_chars2, TEXT_SIZE)
234     clear_buf(newstr1, TEXT_SIZE)
235     clear_buf(newstr2, TEXT_SIZE)
236
237     ret
238
239

```

- **read_name**

- **Описание:** Ввод имени файла с консоли, с удалением символа новой строки (`\n`) в конце.
- **Параметры:**
 - `a0`: Возвращает начало буфера файла без символа новой строки.

- **Особенности:** Работает с фиксированным размером буфера имени файла (`NAME_SIZE`).
- **open_read_file**
 - **Описание:** Открывает файл для чтения, считывает его содержимое в динамически выделенный буфер и завершает чтение, добавляя нулевой символ в конец строки.
 - **Параметры:**
 - `a0`: Возвращает указатель на буфер с содержимым файла.
 - `a1`: Возвращает длину прочитанной строки.
 - **Особенности:** Автоматически расширяет буфер, если содержимое файла превышает начальный размер.
- **save_file**
 - **Описание:** Записывает данные из заданного буфера в файл.
 - **Параметры:**
 - `a1`: Указатель на буфер с текстом для записи.
 - `a2`: Размер текста для записи.
 - **Особенности:** Проверяет корректность открытия файла перед записью. Сообщает об ошибках и завершает выполнение в случае некорректного открытия.
- **strcpy**
 - **Описание:** Копирует строку из одного адреса в другой, включая завершающий символ `\0`.
 - **Параметры:**
 - `a5`: Указатель на исходную строку.
 - `a6`: Указатель на целевую строку.
 - **Особенности:** Работает с побайтным копированием символов до нуля.
- **check_choice**
 - **Описание:** Проверяет пользовательский ввод (`Y` или `N`) и возвращает результат выбора.
 - **Параметры:**
 - `a0`: Возвращает 1 для `Y` (да) и 0 для `N` (нет).
 - **Особенности:** При некорректном вводе повторяет запрос, выводя сообщение об ошибке.
- **choice**
 - **Описание:** Запрашивает у пользователя, выводить ли результаты программы на консоль. При положительном ответе (`Y`) выводит



результаты.

- **Параметры:** Нет.
- **Особенности:** Использует подпрограмму `check_choice` для получения ответа.
- **clear_buffers**
 - **Описание:** Очищает буферы, заполняя их нулями.
 - **Параметры:** Нет.
 - **Особенности:** Использует макрос `clear_buf` для последовательной очистки заданных массивов.

Нная часть этих подпрограмм, код взятый из программ рассмотренных на семинаре и немного измененных. Когда я перехожу в любую из подпрограмм я сохраняю s-регистры на стеке, чтобы оставить их такими, какими они были до этого, чтобы следовать конвенциям.

Тестовое покрытие и автотесты

Были считаны, следующие файлы и получен следующий результат

 example1.txt 14 КБ Изменен: Сегодня, 21:03 Добавить теги... ▼ Основные: Тип: Документ простого текста Размер: 13,855 Б (16 КБ на диске) Где: Macintosh HD ► Пользователи ►	 example2.txt 14 Б Изменен: Сегодня, 21:03 Добавить теги... ▼ Основные: Тип: Документ простого текста Размер: 14 Б (4 КБ на диске) Где: Macintosh HD ► Пользователи ►
---	---

(они будут также прикреплены к работе)

Введите путь до читаемого файла: **** user input : example1.txt
Введите путь до читаемого файла: **** user input : example2.txt

Хотите ли вы вывести результаты работы программы на консоль [Y/N]: **** user input : Y

Вот символы, которые есть в первой строке, но нет во второй:

Lorem ipsudlta,qnxcbvfvf.Ug
EhNTASQID8(){}|\/?

Вот символы, которые есть во второй строке, но нет в первой:
!~

Введите имя для первого файла вывода строки (с расширением .txt): **** user input : 1.txt
Введите имя для второго файла вывода строки (с расширением .txt): **** user input : 2.txt
Хотите ли вы завершить работу программы [Y/N]: **** user input : Y

— program is finished running (0) —

Вот то, что получилось в соответствующих файликах

Тестовая логика точно такая же как и в мейне, но с дополнительным копирование одних строк в другие + присутствует дополнительные выводы строк.

Тестовые запуски состояли из самых разных тестов, где проверялись все возможные случаи работы. Проверка 2 строк без чисел, с числами, спец. символами, одинаковые строки, строка с пустой строкой и их различные комбинации.

```
92 test_logic:
93     push(ra)
94
95     newline
96     newline
97     print_str(test_input_mes)
98     print_str(test_input1)
99     newline
100
101     strcpy(test_input1,file_name)
102     jal open_read_file
103     mv s8 a0          # Сохранение адреса начала буфера.
104     mv s7 a1          # Сохранение величины прочитанной строки.
105
106     # Ввод имени файла с консоли эмулятора
107     print_str(test_input_mes)
108     print_str(test_input2)
109
110     newline
111
112     strcpy(test_input2,file_name)
113     jal open_read_file
114     mv s6 a0          # Сохранение адреса начала буфера.
115     mv s5 a1          # Сохранение величины прочитанной строки.
116
117     create_array_uniq(test_array1, s8, s10) # Макрос для создания массива уникальных символов из строки
118     create_array_uniq(test_array2, s6, s9)  # Макрос для создания массива уникальных символов из строки
119
120     compare(newstr1, test_array1,s10, test_array2,s9) # Макрос для создания строки с разностью символов строк путем сравнения 2 массивов.
121     compare(newstr2, test_array2,s9, test_array1,s10) # Макрос для создания строки с разностью символов строк путем сравнения 2 массивов.
122
123     # Вывод в консоль пути результатов тестов
124     print_str(test_output_mes)
125     print_str(test_output1)
126     newline
127
128     strcpy(test_output1,file_name)
129
130     la a1 newstr1      # Передаю параметры (a1 – адрес начала буфера строки, a2 – размер) в подпрограмму
131     mv a2 s10
132     la t6 save_file    # Подпрограмма для сохранения файлов
133     jalr t6
134
135     # Вывод в консоль пути результатов тестов
136     print_str(test_output_mes)
137     print_str(test_output2)
138     newline
139
140     strcpy(test_output2,file_name)
141
142     la a1 newstr2      # Передаю параметры (a1 – адрес начала буфера строки, a2 – размер) в подпрограмму
143     mv a2 s9
144     la t6 save_file    # Подпрограмма для сохранения файлов
145     jalr t6
146
147     j clear_bufs
148
```

Проведя тестовые запуски, видим, что программа все корректно отработала и завершилась без ошибок, результаты тестов можно посмотреть в соответствующей папке.

Итог

Таким образом, работа сделана на 10 и соблюдает все требования необходимы для каждого отдельного подпункта. Например, она разбита на отдельные единицы компиляции, у макросов присутствует отдельная библиотека, есть авто и просто тестирование, обрабатываются файлы до 10кб при размере буфера на 512 байт.

Желаю проверяющему отличного дня и хорошего настроения!!!