Зенин Вадим ИДЗ по АВС. Вариант -35

Отчет на 6-7 баллов.

Отчет выполнен сразу на 6-7 баллов, поскольку семинарист разрешил так сделать.

Мой вариант - 35.

35. Сформировать массив B из элементов массива A сгруппировов положительные элементы массива в начале массива , нулевые в середине, а отрицательные — в конце.

Macrolib

В этом блоке показаны различные макросы используемые далее в программе.

```
2  # Печать содержимого регистра, если там храниться целочисленное значение.
3  .macro print_int (%x)
4          li a7, 1
5          mv a0, %x
6          ecall
7  .end_macro
8
9  # Ввод целого числа с консоли в указанный регистр,
10 # исключая регистр a0
11 .macro read_int(%x)
12    push (a0)
13    li a7, 5
14    ecall
15    mv %x, a0
16    pop  (a0)
17 .end_macro
18
19 # Печать строки передаваемой в макро.
20 .macro print_str (%x)
21    .data
22 str:
23    .asciz %x
24    .text
25    push (a0)
26    li a7, 4
27    la a0, str
28    ecall
29    pop  (a0)
30    .end_macro
31
```

```
32  # Печать символа передаваемой в макро.
33  .macro print_char(%x)
34     push (a0)
35     li a7, 11
36     li a0, %x
37     ecall
38     pop  (a0)
39     .end_macro
40
41  # Перевод строки.
42  .macro newline
43     print_char('\n')
44     .end_macro
45
46  # Завершение программы
47  .macro exit
48      li a7, 10
49      ecall
50  .end_macro
51
52  # Сохранение заданного регистра на стеке
53  .macro push(%x)
54          addi    sp, sp, -4
55          sw      %x, (sp)
56  .end_macro
57
58  # Выталкивание значения с вершины стека в регистр
59  .macro pop(%x)
60          lw      %x, (sp)
61          addi    sp, sp, 4
62  .end_macro
```

```
62
63   # Вывод поэлементно массива с адресом начала в "%x".
64   .macro  print_array(%x, %size)
65           push(a2)
66           push(a3)
67           mv a3, zero
68
69   loop_print:
70           lw a2 (%x)
71           print_int(a2)
72           print_char(' ')
73           addi %x, %x, 4
74           addi a3, a3, 1
75           bgt %size, a3, loop_print
76           newline
77
78           pop(a3)
79           pop(a2)
80   .end_macro
81
```

Блок .data

На этом скриншоте показаны блок .data

```
.data


        A_array: .space  40             # Поскольку 1 целое число храниться в 4 байта, следовательно для хранения 10 чисел хватит 40 байт.
        B_array: .space  40

```

Блок main

```
.text
main:
        la t0 A_array                                   # Передаю адрес массива A в регистр t0
        la t1 B_array                                   # Передаю адрес массива B в регистр t1
        addi t3, t3, 10                                 # Максимальный размер массива
        la t2 input                                     # Возвращаемое занчение (размер массива) передается в регистр a1
        jalr  t2
        la t2 check_size_array                          # Переходим в подпрограмму для проверки введенного числа (параметр, находящийсяв a1).
        jalr t2
        la t2 prep_loop_input                           # Поэлементное заполнение массива (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива)
        jalr  t2
        la t2 algorithm                                 # Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A,t1 – адрес массив B)
        jalr  t2
        print_str("Получившийся массив B:")
        newline
        print_array(t1, a1)
        exit

```

Сначала загружаю в регистры t0 и t1, адреса начала массивов A и B. В t3 передаю максимальное значение размера массивов, который далее буду использовать для проверки вводимых пользователем данных.
Далее программа состоит из последовательных переходов в подпрограммы использую jalr. После выполнения подпрограмм запускаю макрос, который выводить на экран элементы массива.
В самом конце запускаю макрос для завершения программы.

Ввод данных

На данном скриншоте показана реализация процесса запроса размера массивов у пользователя.
Для начала я сохраняю ra на стеке, чтобы после корректно вернуться.
Вывожу на экран пользователю подсказку и считываю его число.
После выхожу из подпрограммы и далее захожу в следующую для проверки введенного значения и его возможного изменения, если оно некорректно (выходит за рамки этого интервала (0; 10]. В эту подпрограмму передаю параметр - размер массива, через регистр a1.

```

# Подпрограмма ввода размера массива через консоль.
input:

        push(ra)                                        # При заходе в подпрограмму сразу сохраняем на стеке ra.
        print_str("Введите количество элементов в массиве (1-10):")
        read_int(a1)                                    # Размер массива в a1
        newline
        pop(ra)                                         # Возвращаем ra из стека.
        ret                                             # Выходим из подпрограммы после проверки значения рзмера массива по ra.

# Подпрограмма для проверки числа, введенного с консоли.
check_size_array:
        push(ra)
        bgt a1, t3, error_size_array                    # Проверяем больше ли значание размера массива в регистре a1, чем константа (10) в t3, введенная ранее.
        blez  a1, error_size_array                      # Проверяю больше ли значение 0.
        pop(ra)                                         # Возвращаем ra из стека.
        ret                                             # Выходим из подпрограммы после проверки значения рзмера массива по ra.

# Подпрограмма, которая выводит сообщение о неверном значении размера массива и заставляет пользователя поменять значение.
error_size_array:
        print_str("Некорректный размер массива введите число от 1 до 10:")
        read_int(a1)                                    # Считываю значение пользователя.
        newline
        j check_size_array                              # Отправляю значение на новую проверку.

```

Далее идет основной цикл, который заполняет массив A введенными целыми числа. В a1 хранится параметр, равный размеру массива.

Основной алгоритм

> Алгоритм состоит из 3 циклов, каждый из которых заполняет массив B данными из массива A, положительными, нулями и отрицательными числами соответственно.

> Перед началом алгоритма запоминаем ra и загружаем в регистр t0 адрес массива A. Цикл бежит по всем элементам из A, и если он положительный, то добавляет в массив B. Аналогичным образом работают и остальные 2 цикла. После 1 цикла обновляем итератор и адрес массива A.

```
# Основной алгоритм, по которому происходит заполнение массива B. Алгоритм содержит 3 цикла для заполнения положительных, нулевых и отрицательных элементов соответственно.
algorithm:
        push(ra)                                                # Кладем ra на стек.
        la t0 A_array                                           # Загрузили в t0 начало массива A.

# Цикл для заполенинния массива положительными числами.
# Бежим по всему массиву и проверяем каждое число на положительность.
loop_for_positive:
        lw a2 (t0)                                              # Загружаем число по адресу t0 в регистр a2.
        bgtz a2, push_positive                                  # Проверяем позитивное ли оно.
        addi t0, t0, 4                                          # Обновляем адрес следующего числа и
        addi a3, a3, 1                                          # Итератор.
        bgt a1, a3, loop_for_positive
        j reset_for_null                                        # Если прошли весь массив, переходим к подготовлению массива и итератора для аналогичного цикла для нулевых значений.

# Если число положительное, передаем его в начало массива B по регистру t1.
push_positive:
        sw a2 (t1)
        addi t1, t1, 4
        addi t0, t0, 4
        addi a3, a3, 1
        bgt a1, a3, loop_for_positive                           # Возвращаемся дальше в цикл.

# Обновляем итератор и регистр массива.
reset_for_null:
        mv a3, zero
        la t0 A_array

```

Цикл для нулей.

```

# Аналогичный цикл, что и для положительных чисел, однако проверяет числа на равенство нулю.
loop_for_null:
        lw a2 (t0)
        beqz a2, push_null
        addi t0, t0, 4
        addi a3, a3, 1                                          # Итератор
        bgt a1, a3, loop_for_null
        j reset_for_negative

# Заполнение массива B нулевыми элементами.
push_null:
        sw a2 (t1)
        addi t1, t1, 4
        addi t0, t0, 4
        addi a3, a3, 1
        bgt a1, a3, loop_for_null

# Обновляем итератор и регистр массива.
reset_for_negative:
        mv a3, zero
        la t0 A_array

```

Цикл для отрицательных чисел.

```
# Аналогичный цикл, что и для положительных чисел, однако проверяет числа на отрицательность.
loop_for_negative:
        lw a2 (t0)
        bltz a2, push_negative
        addi t0, t0, 4
        addi a3, a3, 1                                          # Итератор
        bgt a1, a3, loop_for_negative
        j end_alghoritm

# Заполнение массива B отрицательными элементами.
push_negative:
        sw a2 (t1)
        addi t1, t1, 4
        addi t0, t0, 4
        addi a3, a3, 1
        bgt a1, a3, loop_for_negative
# Конец алгоритма возвращаем все знания из стека и обновляем в регистрах t0 и t1 адреса массивов.
end_alghoritm:
        la t0 A_array                                           # Загрузили в T0 начало массива A
        la t1 B_array                                           # Загрузили в T1 начало массива B
        pop(ra)
        ret

```

> После окончания алгоритма обновляем регистры с адресами массивов и возвращаем ra со стека.

Вывод данных

Вывод данных массива B осуществляется при помощи данного макрос.

```
64  # Вывод поэлементно массива с адресом начала в "%x".
65  .macro  print_array(%x)
66
67          push(ra)
68          push(a2)
69          push(t3)
70          add t3, zero, zero
71          print_str("Получившийся массив B:")
72          newline
73
74  loop_print:
75          lw a2 (%x)
76          print_int(a2)
77          print_char(' ')
78          addi %x, %x, 4
79          addi t3, t3, 1
80          bgt a1, t3, loop_print
81          newline
82
83          pop(t3)
84          pop(a2)
85          pop(ra)
86          ret
87  .end_macro
```

Тестовое покрытие

- Ввод размера массива пользователем, программа корректно запрашивает у пользователя повторный ввод

```
Введите количество элементов в массиве (1-10):**** user input : 0

Некорректный размер массива введите число от 1 до 10:**** user input : -453

Некорректный размер массива введите число от 1 до 10:**** user input : 254

Некорректный размер массива введите число от 1 до 10:**** user input : 10

Введите элемент массива:**** user input : 1
```

- Массив с различними элементами заполняется корректно.

```
Введите количество элементов в массиве (1-10):**** user input : 5

Введите элемент массива:**** user input : 0

Введите элемент массива:**** user input : 0

Введите элемент массива:**** user input : 0

Введите элемент массива:**** user input : -52

Введите элемент массива:**** user input : -52

Получившийся массив B:
0 0 0 -52 -52
```

```
Некорректный размер массива введите число от 1 до 10:**** user input : 10

Введите элемент массива:**** user input : 1

Введите элемент массива:**** user input : -4

Введите элемент массива:**** user input : 2

Введите элемент массива:**** user input : 0

Введите элемент массива:**** user input : 0

Введите элемент массива:**** user input : 4

Введите элемент массива:**** user input : 6

Введите элемент массива:**** user input : -4

Введите элемент массива:**** user input : -525252

Введите элемент массива:**** user input : 52

Получившийся массив B:
1 2 4 6 52 0 0 -4 -4 -525252
```

```
Введите количество элементов в массиве (1-10):**** user input : 8

Введите элемент массива:**** user input : 1

Введите элемент массива:**** user input : 2

Введите элемент массива:**** user input : 3

Введите элемент массива:**** user input : 4

Введите элемент массива:**** user input : -52

Введите элемент массива:**** user input : -52

Введите элемент массива:**** user input : -52

Введите элемент массива:**** user input : -52

Получившийся массив B:
1 2 3 4 -52 -52 -52 -52
```

```
Введите количество элементов в массиве (1-10):**** user input : 6

Введите элемент массива:**** user input : 1

Введите элемент массива:**** user input : 2

Введите элемент массива:**** user input : 3

Введите элемент массива:**** user input : 0

Введите элемент массива:**** user input : 0

Введите элемент массива:**** user input : 6

Получившийся массив B:
1 2 3 6 0 0
```

- Массивы с элементами одного знака или 0 просто копируются.

```
Введите количество элементов в массиве (1-10):**** user input : 5

Введите элемент массива:**** user input : 1

Введите элемент массива:**** user input : 2

Введите элемент массива:**** user input : 4

Введите элемент массива:**** user input : 3

Введите элемент массива:**** user input : 8

Получившийся массив B:
1 2 4 3 8
```

```
Введите количество элементов в массиве (1-10):**** user input : 4

Введите элемент массива:**** user input : 0

Введите элемент массива:**** user input : 0

Введите элемент массива:**** user input : 0

Введите элемент массива:**** user input : 0

Получившийся массив B:
0 0 0 0
```

```
Введите количество элементов в массиве (1-10):**** user input : 5

Введите элемент массива:**** user input : -9

Введите элемент массива:**** user input : -8

Введите элемент массива:**** user input : -7

Введите элемент массива:**** user input : -6

Введите элемент массива:**** user input : -5

Получившийся массив B:
-9 -8 -7 -6 -5
```

Отчет на 8 баллов.

- Разработанные подпрограммы должны поддерживать многократное использование с различными наборами исходных данных, включая возможность подключения различных исходных и результирующих массивов. То есть, поддерживать работу с формальными и фактическими параметрами.

Разработанные мной подпрограммы поддерживают работу с формальными и фактическими параметрами.

- Рассмотрим первую для ввода данных.
  Она позволяет вводить любое целое число с клавиатуры, те универсальна.
- Рассмотрим подпрограмму для проверки.
  Она позволяет определить принадлежит ли параметр диапазону [1:10].
- Рассмотрим ввод данных.
  Он также позволяет вводить любые целые числа в наш массив.
- Основной алгоритм.
  Как показало тестовое покрытие он, независимо от массива A, реализует составление массива B по заданным в задании правилам.

- Вывод данных.

  Данный макрос позволяет вывести на экран любой массив, адрес которого передают.

  Вывод все подпрограммы универсальны и соответствуют требованию

Тестовая программа

- Реализовать автоматизированное тестирование за счет создания **дополнительной тестовой программы**, осуществляющей прогон подпрограммы обработки массивов с различными тестовыми данными (вместо ввода данных). Осуществить прогон тестов обеспечивающих покрытие различных ситуаций. Тестовые данные можно формировать в различных исходных массивах.

Всего будет 7 блоков тестов:

1. Массив состоит из положительных, отрицательных и нулей
2. из положительных и нулей
3. из отрицательных и нулей
4. их положительных и отрицательных
5. только из положительных
6. только из нулей
7. только из отрицательных

   В последних трех случаях массив должен просто копироваться.
8. Дополнили блок .data. Добавив туда массивы с различным количеством данных для каждого тестового блока.

```
4  .data
5
6
7          A_array: .space  40            # Поскольку 1 целое число храниться в 4 байта, следовательно для хранения 10 чисел хватит 40 байт.
8          B_array: .space  40
9          test_array_1: .word 1,-3, 5, 0, 7, -5, 0, 0,-52               # Массив, состоящий из положительных, отрицательных и нулей
10         test_array_2: .word 1, 3, 5, 0, 0,  52, 0, 0                  # Массив, состоящий из положительных и нулей
11         test_array_3: .word 0, 0, -52, 0, 0, -52, 0, 0,-52            # Массив, состоящий из отрицательных и нулей
12         test_array_4: .word 52, 7, -25, 2345, 5252,-52                # Массив, состоящий из положительных и отрицательных чисел
13         test_array_5: .word 1,2,5,3,8                                 # Массив, состоящий из положительных чисел
14         test_array_6: .word -1,-3, -5,                                # Массив, состоящий отрицательных чисел
15         test_array_7: .word 0                                         # Массив, состоящий из нулей
16
```

   Добавим также в регистр t4 значение -52. Пусть если пользователь ввел его, то будет запускаться автотестирование. Добавим проверку в подпрограмме.

```
60 # Подпрограмма для проверки числа, введенного с консоли.
61 check_size_array:
62         push(ra)
63         beq a1, t4, testing                                           # Проверка на запуска автотестирования.
64         bgt a1, t3, error_size_array                                  # Проверяем больше ли значание размера массива в регистре a1, чем константа (10) в t3, введенная ранее.
65         blez  a1, error_size_array                                    # Проверяю больше ли значение 0.
66         pop(ra)                                                       # Возвращаем ra из стека.
67         ret                                                           # Выходим из подпрограммы после проверки значения рзмера массива по ra.
68
```

9. Напишем макрос для копирования массива для тестирования в массива A.

```
# Копирование 1 массива %x в регистр %array_copy, size - размер, передаваемого массива.
.macro copy_array(%array_copy, %x,%size)

.text
        push(ra)
        push(t6)
        mv t5, %x
        mv a3, zero                                     # Обнуляем итератор
loop:
        lw t6, (t5)                                     # Загрузить текущий элемент из исходного массива в t6
        sw t6, (%array_copy)                            # Сохранить элемент в A_array
        addi t5, t5, 4                                  # Перейти к следующему элементу в исходном массиве
        addi %array_copy, %array_copy, 4                # Перейти к следующему элементу в A_array
        addi a3, a3, 1                                  # Итератор
        bgt %size, a3, loop                             # Повторяем цикл пока итератор меньше размера массива
end_copy:
        mv a3, zero                                     # Обнуляем итератор
        pop(t6)
        pop(ra)

.end_macro

```

10. Далее напишем группу схожих последовательных тестов, с различными входными данными.

```
testing:
        test1:
                la t6, test_array_1                     # Загружаем тестовый массив в регистр t5
                li a1, 9                                # Размер тестового массива
                copy_array(t0,t6,a1)                    # Макро для копирования массива
                la t2 algorithm                         # Основной алгоритм заполнения массива B (параметры: a1 - размер массивов, a3 - итератор, t0 - адрес массива A,t1 - адрес массив B)
                jalr  t2
                print_str("Массив, переданный в программу: ")
                print_array(t6,a1)
                print_str("Получившийся массив B:")
                print_array(t1, a1)                     # Ввод массива.
                newline
                j test2
        test2:
                la t1, B_array
                la t6, test_array_2                     # Загружаем тестовый массив в регистр t5
                li a1, 8                                # Размер тестового массива
                copy_array(t0,t6,a1)                    # Макро для копирования массива
                la t2 algorithm                         # Основной алгоритм заполнения массива B (параметры: a1 - размер массивов, a3 - итератор, t0 - адрес массива A,t1 - адрес массив B)
                jalr  t2
                print_str("Массив, переданный в программу: ")
                print_array(t6,a1)
                print_str("Получившийся массив B:")
                print_array(t1, a1)                     # Ввод массива.
                newline
                j test3
        test3:
                la t1, B_array
                la t6, test_array_3                     # Загружаем тестовый массив в регистр t5
                li a1, 9                                # Размер тестового массива
                copy_array(t0,t6,a1)                    # Макро для копирования массива
                la t2 algorithm                         # Основной алгоритм заполнения массива B (параметры: a1 - размер массивов, a3 - итератор, t0 - адрес массива A,t1 - адрес массив B)
```

Сначала в регистр t6 загружаю тестовый массив. Потом в a1 загружаю размер этого массива. После использую макрос для копирования массива из t6 в t0, размером a1. Потом с этими данными запускаю алгоритм, он выполняется. Дальше вывожу исходный массив и получившийся. После прохождения всех тестов завершаю программу.
Результат прохождения тестов:![[Pasted image 20241019003849.png]

Отчет на 9

Макросы

Все макросы, используемые в программе, и их реализация предоставлены выше. В этом пункте я вкратце опишу их функционал.

- print_int (%x) - выводит на экран число из регистра %x.
- read_int(%x) - считывает число в регистр %x, исключая a0.
- print_str (%x) - выводит на экран строку %x.
- print_char(%x) - выводит на экран символ %x.
- newline - переводит строку.

- exit - вызывает системный вызов 10 - завершение программы.
- push(%x) - сохранение регистра %x на стеке.
- pop(%x) - "выталкивание" значения с вершины стека в регистр %x.
- print_array(%x, %size) - вывод поэлементно массива %x размером %size на экран.
- copy_array(%array_copy, %x,%size) - копирование массива %x размером %size в массив %array_copy.

Отчет на 10

Разобьем программу по разным файлам.

main.asm

Файлик с запуском основной программы.

```
.include "macrolib.asm"
.globl A_array, B_array
.data


        A_array: .space  40             # Поскольку 1 целое число храниться в 4 байта, следовательно для хранения 10 чисел хватит 40 байт.
        B_array: .space  40

.text
main:
        la t0 A_array                                           # Передаю адрес массива A в регистр t0
        la t1 B_array                                           # Передаю адрес массива B в регистр t1
        addi t3, t3, 10                                         # Максимальный размер массива
        addi t4, t4, -52                                        # Константа для запуска автоматического тестирования.
        la t2 input                                             # Возвращаемое занчение (размер массива) передается в регистр a1
        jalr  t2
        la t2 check_size_array                                  # Переходим в подпрограмму для проверки введенного числа (параметр, находящийсяв a1).
        jalr t2
        la t2 prep_loop_input                                   # Поэлементное заполнение массива (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива)
        jalr  t2
        la t2 algorithm                                         # Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A,t1 – адрес массив B)
        jalr  t2
        print_str("Получившийся массив B:")
        newline
        print_array(t1, a1)
        exit

```

input.asm

Файлик с подпрограммой ввода данных в массив.

```
.globl input, check_size_array, prep_loop_input

.text
# Подпрограмма ввода размера массива через консоль.
input:

        push(ra)                                                # При заходе в подпрограмму сразу сохраняем на стеке ra.
        print_str("Введите количество элементов в массиве (1-10, введите: -52 для автоматического тестирования):")
        read_int(a1)                                            # Размер массива в a1
        newline
        pop(ra)                                                 # Возвращаем ra из стека.
        ret                                                     # Выходим из подпрограммы после проверки значения рзмера массива по ra.


# Подпрограмма для проверки числа, введенного с консоли.
check_size_array:
        push(ra)
        beq a1, t4, tests                                       # Проверка на запуска автотестирования.
        bgt a1, t3, error_size_array                            # Проверяем больше ли значание размера массива в регистре a1, чем константа (10) в t3, введенная ранее.
        blez  a1, error_size_array                              # Проверяю больше ли значение 0.
        pop(ra)                                                 # Возвращаем ra из стека.
        ret                                                     # Выходим из подпрограммы после проверки значения рзмера массива по ra.

# Подпрограмма, которая выводит сообщение о неверном значении размера массива и заставляет пользователя поменять значение.
error_size_array:
        print_str("Некорректный размер массива введите число от 1 до 10:")
        read_int(a1)                                            # Считываю значение пользователя.
        newline
        j check_size_array                                      # Отправляю значение на новую проверку.


# Подготовка перед входом в цикл заполенинния массива.
prep_loop_input:
        push(ra)                                                # При заходе в подпрограмму сразу сохраняем на стеке ra.

# Основной цикл, который заполняет массив A.
loop_input:

        print_str("Введите элемент массива:")
        read_int(a2)                                            # Считываем элемент в a2.
        newline

        sw a2, (t0)                                             # Загружаем значение a2 по адресу начала массива A.
        addi a3, a3, 1
        addi t0, t0, 4                                          # Сдвигаем адрес в регистре t0 для следующего элемента.
        bgt a1, a3 loop_input                                   # Проверка на выход из цикла.
        la t0 A_array                                           # Загрузили в T0 начало массива A.
        mv a3, zero                                             # Обнуляю итератор для следующего использования.
        pop(ra)                                                 # Возвращаем ra из стека.
        ret                                                     # Выходим из подпрограммы после проверки значения рзмера массива по ra.
```

algorithm.asm

Файлик, который содержит основной алгоритм создания массива B.

```
.include "macrolib.asm"
.globl algorithm


.text
# Основной алгоритм, по которому происходит заполнение массива B. Алгоритм содержит 3 цикла для заполнения положительных, нулевых и отрицательных элементов соответственно.
algorithm:
        push(ra)                                                # Кладем ra на стек.
        la t0 A_array                                           # Загрузили в t0 начало массива A.

# Цикл для заполенинния массива положительными числами.
# Бежим по всему массиву и проверяем каждое число на положительность.
loop_for_positive:
        lw a2 (t0)                                              # Загружаем число по адресу t0 в регистр a2.
        bgtz a2, push_positive                                  # Проверяем позитивное ли оно.
        addi t0, t0, 4                                          # Обновляем адрес следующего числа и
        addi a3, a3, 1                                          # Итератор.
        bgt a1, a3, loop_for_positive
        j reset_for_null                                        # Если прошли весь массив, переходим к подготовлению массива и итератора для аналогичного цикла для нулевых значений.

# Если число положительное, передаем его в начало массива B по регистру t1.
push_positive:
        sw a2 (t1)
        addi t1, t1, 4
        addi t0, t0, 4
        addi a3, a3, 1
        bgt a1, a3, loop_for_positive                           # Возвращаемся дальше в цикл.

# Обновляем итератор и регистр массива.
reset_for_null:
        mv a3, zero
        la t0 A_array
```

```
# Аналогичный цикл, что и для положительных чисел, однако проверяет числа на равенство нулю.
loop_for_null:
        lw a2 (t0)
        beqz a2, push_null
        addi t0, t0, 4
        addi a3, a3, 1                                          # Итератор
        bgt a1, a3, loop_for_null
        j reset_for_negative

# Заполнение массива B нулевыми элементами.
push_null:
        sw a2 (t1)
        addi t1, t1, 4
        addi t0, t0, 4
        addi a3, a3, 1
        bgt a1, a3, loop_for_null

# Обновляем итератор и регистр массива.
reset_for_negative:
        mv a3, zero
        la t0 A_array

# Аналогичный цикл, что и для положительных чисел, однако проверяет числа на отрицательность.
loop_for_negative:
        lw a2 (t0)
        bltz a2, push_negative
        addi t0, t0, 4
        addi a3, a3, 1                                          # Итератор
        bgt a1, a3, loop_for_negative
        j end_alghoritm

# Заполнение массива B отрицательными элементами.
push_negative:
        sw a2 (t1)
        addi t1, t1, 4
        addi t0, t0, 4
        addi a3, a3, 1
        bgt a1, a3, loop_for_negative
# Конец алгоритма возвращаем все знания из стека и обновляем в регистрах t0 и t1 адреса массивов.
end_alghoritm:
        la t0 A_array                                           # Загрузили в T0 начало массива A
        la t1 B_array                                           # Загрузили в T1 начало массива B
        pop(ra)
        ret

```

tests.asm

Файлик, содержащий реализацию тестов.

```
.include "macrolib.asm"
.globl tests


.data

        test_array_1: .word 1,-3, 5, 0, 7, -5, 0, 0,-52                # Массив, состоящий из положительных, отрицательных и нулей
        test_array_2: .word 1, 3, 5, 0, 0,  52, 0, 0                   # Массив, состоящий из положительных и нулей
        test_array_3: .word 0, 0, -52, 0, 0, -52, 0, 0,-52             # Массив, состоящий из отрицательных и нулей
        test_array_4: .word 52, 7, -25, 2345, 5252,-52                 # Массив, состоящий из положительных и отрицательных чисел
        test_array_5: .word 1,2,5,3,8                                  # Массив, состоящий из положительных чисел
        test_array_6: .word -1,-3, -5                                  # Массив, состоящий отрицательных чисел
        test_array_7: .word 0                                          # Массив, состоящий из нулей

.text
tests:
        test1:
                la t6, test_array_1                                    # Загружаем тестовый массив в регистр t5
                li a1, 9                                               # Размер тестового массива
                copy_array(t0,t6,a1)                                   # Макро для копирования массива
                la t2 algorithm                                        # Основной алгоритм заполнения массива B (параметры: a1 - размер массивов, a3 - итератор, t0 - адрес массива A,t1 - адрес массив B)
                jalr  t2
                print_str("Массив, переданный в программу: ")
                print_array(t6,a1)
                print_str("Получившийся массив B:")
                print_array(t1, a1)                                    # Ввод массива.
                newline
                j test2
        test2:
                la t1, B_array
                la t6, test_array_2                                    # Загружаем тестовый массив в регистр t5
                li a1, 8                                               # Размер тестового массива
                copy_array(t0,t6,a1)                                   # Макро для копирования массива
                la t2 algorithm                                        # Основной алгоритм заполнения массива B (параметры: a1 - размер массивов, a3 - итератор, t0 - адрес массива A,t1 - адрес массив B)
                jalr  t2
                print_str("Массив, переданный в программу: ")
                print_array(t6,a1)
                print_str("Получившийся массив B:")
                print_array(t1, a1)                                    # Ввод массива.
                newline
                j test3
```

```
test3:
        la t1, B_array
        la t6, test_array_3                     # Загружаем тестовый массив в регистр t5
        li a1, 9                                # Размер тестового массива
        copy_array(t0,t6,a1)                    # Макро для копирования массива
        la t2 algorithm                         # Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A,t1 – адрес массив B)
        jalr  t2
        print_str("Массив, переданный в программу: ")
        print_array(t6,a1)
        print_str("Получившийся массив B:")
        print_array(t1, a1)                     # Ввод массива.
        newline
        j test4
test4:
        la t1, B_array
        la t6, test_array_4                     # Загружаем тестовый массив в регистр t5
        li a1, 6                                # Размер тестового массива
        copy_array(t0,t6,a1)                    # Макро для копирования массива
        la t2 algorithm                         # Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A,t1 – адрес массив B)
        jalr  t2
        print_str("Массив, переданный в программу: ")
        print_array(t6,a1)
        print_str("Получившийся массив B:")
        print_array(t1, a1)                     # Ввод массива.
        newline
        j test5
test5:
        la t1, B_array
        la t6, test_array_5                     # Загружаем тестовый массив в регистр t5
        li a1, 5                                # Размер тестового массива
        copy_array(t0,t6,a1)                    # Макро для копирования массива
        la t2 algorithm                         # Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A,t1 – адрес массив B)
        jalr  t2
        print_str("Массив, переданный в программу: ")
        print_array(t6,a1)
        print_str("Получившийся массив B:")
        print_array(t1, a1)                     # Ввод массива.
        newline
        j test6

test6:
        la t1, B_array
        la t6, test_array_6                     # Загружаем тестовый массив в регистр t5
        li a1, 3                                # Размер тестового массива
        copy_array(t0,t6,a1)                    # Макро для копирования массива
        la t2 algorithm                         # Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A,t1 – адрес массив B)
        jalr  t2
        print_str("Массив, переданный в программу: ")
        print_array(t6,a1)
        print_str("Получившийся массив B:")
        print_array(t1, a1)                     # Ввод массива.
        newline
        j test7
test7:
        la t1, B_array
        la t6, test_array_7                     # Загружаем тестовый массив в регистр t5
        li a1, 1                                # Размер тестового массива
        copy_array(t0,t6,a1)                    # Макро для копирования массива
        la t2 algorithm                         # Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A,t1 – адрес массив B)
        jalr  t2
        print_str("Массив, переданный в программу: ")
        print_array(t6,a1)
        print_str("Получившийся массив B:")
        print_array(t1, a1)                     # Ввод массива.
        newline
        exit
```

macrolib.asm

Файлик, содержащий библиотеку макросов.

```
2  # Печать содержимого регистра, если там храниться целочисленное значение.
3  .macro print_int (%x)
4          li a7, 1
5          mv a0, %x
6          ecall
7  .end_macro
8
9  # Ввод целого числа с консоли в указанный регистр,
10 # исключая регистр a0
11 .macro read_int(%x)
12    push (a0)
13    li a7, 5
14    ecall
15    mv %x, a0
16    pop  (a0)
17 .end_macro
18
19 # Печать строки передаваемой в макро.
20 .macro print_str (%x)
21    .data
22 str:
23    .asciz %x
24    .text
25    push (a0)
26    li a7, 4
27    la a0, str
28    ecall
29    pop  (a0)
30    .end_macro
31
32 # Печать символа передаваемой в макро.
33 .macro print_char(%x)
34    push (a0)
35    li a7, 11
36    li a0, %x
37    ecall
38    pop  (a0)
39    .end_macro
40
41 # Перевод строки.
42 .macro newline
43    print_char('\n')
44    .end_macro
45
46 # Завершение программы
47 .macro exit
48     li a7, 10
49     ecall
50 .end_macro
51
```

```
.macro push(%x)
        addi    sp, sp, -4
        sw      %x, (sp)
.end_macro

# Выталкивание значения с вершины стека в регистр
.macro pop(%x)
        lw      %x, (sp)
        addi    sp, sp, 4
.end_macro

# Вывод поэлементно массива с адресом начала в "%x".
.macro  print_array(%x, %size)
        push(a2)
        push(a3)
        mv a3, zero

loop_print:
        lw a2 (%x)
        print_int(a2)
        print_char(' ')
        addi %x, %x, 4
        addi a3, a3, 1
        bgt %size, a3, loop_print
        newline

        pop(a3)
        pop(a2)
.end_macro

# Копирование 1 массива %x в регистр %array_copy, size - размер, передаваемого массива.
.macro copy_array(%array_copy, %x,%size)

.text
        push(ra)
        push(t6)
        mv t5, %x
        mv a3, zero                                             # Обнуляем итератор
loop:
        lw t6, (t5)                                             # Загрузить текущий элемент из исходного массива в t6
        sw t6, (%array_copy)                                    # Сохранить элемент в A_array
        addi t5, t5, 4                                          # Перейти к следующему элементу в исходном массиве
        addi %array_copy, %array_copy, 4                        # Перейти к следующему элементу в A_array
        addi a3, a3, 1                                          # Итератор
        bgt %size, a3, loop                                     # Повторяем цикл пока итератор меньше размера массива
end_copy:
        mv a3, zero                                             # Обнуляем итератор
        pop(t6)
        pop(ra)

.end_macro
```