

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 по-ложительные элементы массива в начале массива , нулевые в се-редине, 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
```

```

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

```
3
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
```

Блок main

```
16 .text
17 main:
18     la t0 A_array          # Передаю адрес массива A в регистр t0
19     la t1 B_array          # Передаю адрес массива B в регистр t1
20     addi t3, t3, 10        # Максимальный размер массива
21     la t2 input            # Возвращаемое значение (размер массива) передается в регистр a1
22     jalr t2                # Переходим в подпрограмму для проверки введенного числа (параметр, находящийся в a1).
23     la t2 check_size_array
24     jalr t2                # Поэлементное заполнение массива (параметры: a1 - размер массивов, a3 - итератор, t0 - адрес массива)
25     la t2 prep_loop_input
26     jalr t2                # Основной алгоритм заполнения массива B (параметры: a1 - размер массивов, a3 - итератор, t0 - адрес массива A, t1 - адрес массив B)
27     la t2 algorithm
28     jalr t2
29     print_str("Получившийся массив B:")
30     newline
31     print_array(t1, a1)
32     exit
33
```

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.

```
31
32 # Подпрограмма ввода размера массива через консоль.
33 input:
34
35     push(ra)                # При заходе в подпрограмму сразу сохраняю на стеке ra.
36     print_str("Введите количество элементов в массиве (1-10):")
37     read_int(a1)            # Размер массива в a1
38     newline
39     pop(ra)                 # Возвращаю ra из стека.
40     ret                     # Выходим из подпрограммы после проверки значения размера массива по ra.
41
42 # Подпрограмма для проверки числа, введенного с консоли.
43 check_size_array:
44     push(ra)
45     bgt a1, t3, error_size_array # Проверяю больше ли значение размера массива в регистре a1, чем константа (10) в t3, введенная ранее.
46     blez a1, error_size_array  # Проверяю больше ли значение 0.
47     pop(ra)                   # Возвращаю ra из стека.
48     ret                       # Выходим из подпрограммы после проверки значения размера массива по ra.
49
50 # Подпрограмма, которая выводит сообщение о неверном значении размера массива и заставляет пользователя поменять значение.
51 error_size_array:
52     print_str("Некорректный размер массива введите число от 1 до 10:")
53     read_int(a1)             # Считываю значение пользователя.
54     newline
55     j check_size_array       # Отправляю значение на новую проверку.
56
```

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

Основной алгоритм

Алгоритм состоит из 3 циклов, каждый из которых заполняет массив В данными из массива А, положительными, нулями и отрицательными числами соответственно.

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

```
79 # Основной алгоритм, по которому происходит заполнение массива В. Алгоритм содержит 3 цикла для заполнения положительных, нулевых и отрицательных элементов соответственно.
80 algorithm:
81     push(ra)                                # Кладем ra на стек.
82     la t0 A_array                            # Загрузили в t0 начало массива А.
83
84 # Цикл для заполнения массива положительными числами.
85 # Бежим по всему массиву и проверяем каждое число на положительность.
86 loop_for_positive:
87     lw a2 (t0)                                # Загружаем число по адресу t0 в регистр a2.
88     bgtz a2, push_positive                    # Проверяем позитивное ли оно.
89     addi t0, t0, 4                             # Обновляем адрес следующего числа и
90     addi a3, a3, 1                             # Итератор.
91     bgt a1, a3, loop_for_positive
92     j reset_for_null                          # Если прошли весь массив, переходим к подготовке массива и итератора для аналогичного цикла для нулевых значений.
93
94 # Если число положительное, передаем его в начало массива В по регистру t1.
95 push_positive:
96     sw a2 (t1)
97     addi t1, t1, 4
98     addi t0, t0, 4
99     addi a3, a3, 1
100    bgt a1, a3, loop_for_positive              # Возвращаемся дальше в цикл.
101
102 # Обновляем итератор и регистр массива.
103 reset_for_null:
104     mv a3, zero
105     la t0 A_array
106
```

Цикл для нулей.

```
106
107 # Аналогичный цикл, что и для положительных чисел, однако проверяет числа на равенство нулю.
108 loop_for_null:
109     lw a2 (t0)
110     beqz a2, push_null
111     addi t0, t0, 4
112     addi a3, a3, 1                             # Итератор
113     bgt a1, a3, loop_for_null
114     j reset_for_negative
115
116 # Заполнение массива В нулевыми элементами.
117 push_null:
118     sw a2 (t1)
119     addi t1, t1, 4
120     addi t0, t0, 4
121     addi a3, a3, 1
122     bgt a1, a3, loop_for_null
123
124 # Обновляем итератор и регистр массива.
125 reset_for_negative:
126     mv a3, zero
127     la t0 A_array
128
```

Цикл для отрицательных чисел.

```
129 # Аналогичный цикл, что и для положительных чисел, однако проверяет числа на отрицательность.
130 loop_for_negative:
131     lw a2 (t0)
132     bltz a2, push_negative
133     addi t0, t0, 4
134     addi a3, a3, 1                             # Итератор
135     bgt a1, a3, loop_for_negative
136     j end_algorithm
137
138 # Заполнение массива В отрицательными элементами.
139 push_negative:
140     sw a2 (t1)
141     addi t1, t1, 4
142     addi t0, t0, 4
143     addi a3, a3, 1
144     bgt a1, a3, loop_for_negative
145
146 # Конец алгоритма возвращаем все значения из стека и обновляем в регистрах t0 и t1 адреса массивов.
147 end_algorithm:
148     la t0 A_array                            # Загрузили в T0 начало массива А
149     pop(ra)                                  # Загрузили в T1 начало массива В
150     ret
151
```

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

Вывод данных

Вывод данных массива В 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 В:")
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 различными 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 В:
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 В:
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 В:
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 В:
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 В:
-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 А, реализует составление массива В 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 запуск автотестирования.
64     bgt a1, t3, error_size_array # Проверим больше ли значение размера массива в регистре a1, чем константа (10) в t3, введенная ранее.
65     blez a1, error_size_array   # Проверю больше ли значение 0.
66     pop(ra)                    # Возвращаем ra из стека.
67     ret                        # Выходим из подпрограммы после проверки значения размера массива по ga.
68

```

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

```
81
82 # Копирование 1 массива %x в регистр %array_copy, size – размер, передаваемого массива.
83 .macro copy_array(%array_copy, %x,%size)
84
85 .text
86     push(ra)
87     push(t6)
88     mv t5, %x
89     mv a3, zero                                # Обнуляем итератор
90 loop:
91     lw t6, (t5)                                # Загрузить текущий элемент из исходного массива в t6
92     sw t6, (%array_copy)                       # Сохранить элемент в A_array
93     addi t5, t5, 4                             # Перейти к следующему элементу в исходном массиве
94     addi %array_copy, %array_copy, 4          # Перейти к следующему элементу в A_array
95     addi a3, a3, 1                             # Итератор
96     bgt %size, a3, loop                       # Повторяем цикл пока итератор меньше размера массива
97 end_copy:
98     mv a3, zero                                # Обнуляем итератор
99     pop(t6)
100    pop(ra)
101
102 .end_macro
103
```

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

```
43 testing:
44     test1:
45         la t6, test_array_1                    # Загружаем тестовый массив в регистр t5
46         li a1, 9                               # Размер тестового массива
47         copy_array(t0,t6,a1)                   # Макро для копирования массива
48         la t2 algorithm                        # Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A,t1 – адрес массив B)
49         jalr t2
50         print_str("Массив, переданный в программу: ")
51         print_array(t6,a1)
52         print_str("Получившийся массив B:")
53         print_array(t1, a1)                    # Ввод массива.
54         newline
55     j test2
56
57     test2:
58         la t1, B_array
59         la t6, test_array_2                    # Загружаем тестовый массив в регистр t5
60         li a1, 8                               # Размер тестового массива
61         copy_array(t0,t6,a1)                   # Макро для копирования массива
62         la t2 algorithm                        # Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A,t1 – адрес массив B)
63         jalr t2
64         print_str("Массив, переданный в программу: ")
65         print_array(t6,a1)
66         print_str("Получившийся массив B:")
67         print_array(t1, a1)                    # Ввод массива.
68         newline
69     j test3
70
71     test3:
72         la t1, B_array
73         la t6, test_array_3                    # Загружаем тестовый массив в регистр t5
74         li a1, 9                               # Размер тестового массива
75         copy_array(t0,t6,a1)                   # Макро для копирования массива
76         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.

```

1  .include "macrolib.asm"
2  .globl A_array, B_array
3  .data
4
5
6      A_array: .space 40          # Поскольку 1 целое число храниться в 4 байта, следовательно для хранения 10 чисел хватит 40 байт.
7      B_array: .space 40
8
9  .text
10 main:
11      la t0, A_array             # Передаю адрес массива A в регистр t0
12      la t1, B_array            # Передаю адрес массива B в регистр t1
13      addi t3, t3, 10            # Максимальный размер массива
14      addi t4, t4, -52           # Константа для запуска автоматического тестирования.
15      la t2, input              # Возвращаемое значение (размер массива) передается в регистр a1
16      jalr t2                    # Возвращаемое значение (размер массива) передается в регистр a1
17      la t2, check_size_array    # Переходим в подпрограмму для проверки введенного числа (параметр, находящийся в a1).
18      jalr t2
19      la t2, prep_loop_input     # Поэлементное заполнение массива (параметры: a1 - размер массивов, a3 - итератор, t0 - адрес массива)
20      jalr t2
21      la t2, algorithm           # Основной алгоритм заполнения массива B (параметры: a1 - размер массивов, a3 - итератор, t0 - адрес массива A, t1 - адрес массив B)
22      jalr t2
23      print_str("Получившийся массив B:")
24      newline
25      print_array(t1, a1)
26      exit
27

```

input.asm

Файлик с подпрограммой ввода данных в массив.

```
2 .globl input, check_size_array, prep_loop_input
3
4 .text
5 # Подпрограмма ввода размера массива через консоль.
6 input:
7
8     push(ra)                                # При заходе в подпрограмму сразу сохраняем на стеке ra.
9     print_str("Введите количество элементов в массиве (1-10, введите: -52 для автоматического тестирования):")
10    read_int(a1)                             # Размер массива в a1
11    newline
12    pop(ra)                                  # Возвращаем ra из стека.
13    ret                                     # Выходим из подпрограммы после проверки значения размера массива по ra.
14
15
16 # Подпрограмма для проверки числа, введенного с консоли.
17 check_size_array:
18     push(ra)
19     beq a1, t4, tests                       # Проверка на запуск автотестирования.
20     bgt a1, t3, error_size_array           # Проверим больше ли значение размера массива в регистре a1, чем константа (10) в t3, введенная ранее.
21     blez a1, error_size_array              # Проверю больше ли значение 0.
22     pop(ra)                                # Возвращаем ra из стека.
23     ret                                    # Выходим из подпрограммы после проверки значения размера массива по ra.
24
25 # Подпрограмма, которая выводит сообщение о неверном значении размера массива и заставляет пользователя поменять значение.
26 error_size_array:
27     print_str("Некорректный размер массива введите число от 1 до 10:")
28     read_int(a1)                           # Считываю значение пользователя.
29     newline
30     j check_size_array                     # Отправляю значение на новую проверку.
31
32
33 # Подготовка перед входом в цикл заполнения массива.
34 prep_loop_input:
35     push(ra)                               # При заходе в подпрограмму сразу сохраняем на стеке ra.
36
37 # Основной цикл, который заполняет массив A.
38 loop_input:
39
40     print_str("Введите элемент массива:")
41     read_int(a2)                           # Считываем элемент в a2.
42     newline
43
44     sw a2, (t0)                            # Загружаем значение a2 по адресу начала массива A.
45     addi a3, a3, 1
46     addi t0, t0, 4                          # Сдвигаем адрес в регистре t0 для следующего элемента.
47     bgt a1, a3, loop_input                 # Проверка на выход из цикла.
48     la t0 A_array                          # Загрузили в t0 начало массива A.
49     mv a3, zero                             # Обнуляю итератор для следующего использования.
50     pop(ra)                                # Возвращаем ra из стека.
51     ret                                    # Выходим из подпрограммы после проверки значения размера массива по ra.
52
53
```

algorithm.asm

Файлик, который содержит основной алгоритм создания массива B.

```
1 .include "macrolib.asm"
2 .globl algorithm
3
4
5 .text
6 # Основной алгоритм, по которому происходит заполнение массива B. Алгоритм содержит 3 цикла для заполнения положительных, нулевых и отрицательных элементов соответственно.
7 algorithm:
8     push(ra)                                # Кладем ra на стек.
9     la t0 A_array                          # Загрузили в t0 начало массива A.
10
11 # Цикл для заполнения массива положительными числами.
12 # Бегим по всему массиву и проверяем каждое число на положительность.
13 loop_for_positive:
14     lw a2 (t0)                             # Загружаем число по адресу t0 в регистр a2.
15     bgtz a2, push_positive                 # Проверяем позитивное ли оно.
16     addi t0, t0, 4                          # Обновляем адрес следующего числа и
17     addi a3, a3, 1                          # Итератор.
18     bgt a1, a3, loop_for_positive
19     j reset_for_null                       # Если прошли весь массив, переходим к подготовке массива и итератора для аналогичного цикла для нулевых значений.
20
21 # Если число положительное, передаем его в начало массива B по регистру t1.
22 push_positive:
23     sw a2 (t1)
24     addi t1, t1, 4
25     addi t0, t0, 4
26     addi a3, a3, 1
27     bgt a1, a3, loop_for_positive          # Возвращаемся дальше в цикл.
28
29 # Обновляем итератор и регистр массива.
30 reset_for_null:
31     mv a3, zero
32     la t0 A_array
33
```

```

34 # Аналогичный цикл, что и для положительных чисел, однако проверяет числа на равенство нулю.
35 loop_for_null:
36     lw a2 (t0)
37     beqz a2, push_null
38     addi t0, t0, 4
39     addi a3, a3, 1
40     bgt a1, a3, loop_for_null
41     j reset_for_negative
42
43 # Заполнение массива B нулевыми элементами.
44 push_null:
45     sw a2 (t1)
46     addi t1, t1, 4
47     addi t0, t0, 4
48     addi a3, a3, 1
49     bgt a1, a3, loop_for_null
50
51 # Обновляем итератор и регистр массива.
52 reset_for_negative:
53     mv a3, zero
54     la t0 A_array
55
56 # Аналогичный цикл, что и для положительных чисел, однако проверяет числа на отрицательность.
57 loop_for_negative:
58     lw a2 (t0)
59     bltz a2, push_negative
60     addi t0, t0, 4
61     addi a3, a3, 1
62     bgt a1, a3, loop_for_negative
63     j end_algorithm
64
65 # Заполнение массива B отрицательными элементами.
66 push_negative:
67     sw a2 (t1)
68     addi t1, t1, 4
69     addi t0, t0, 4
70     addi a3, a3, 1
71     bgt a1, a3, loop_for_negative
72 # Конец алгоритма возвращаем все значения из стека и обновляем в регистрах t0 и t1 адреса массивов.
73 end_algorithm:
74     la t0 A_array
75     la t1 B_array
76     pop(ra)
77     ret
78

```

tests.asm

Файлик, содержащий реализацию тестов.

```

1  .include "macrolib.asm"
2  .globl tests
3
4
5  .data
6
7      test_array_1: .word 1,-3, 5, 0, 7, -5, 0, 0,-52
8      test_array_2: .word 1, 3, 5, 0, 0, 52, 0, 0
9      test_array_3: .word 0, 0, -52, 0, 0, -52, 0, 0,-52
10     test_array_4: .word 52, 7, -25, 2345, 5252,-52
11     test_array_5: .word 1,2,5,3,8
12     test_array_6: .word -1,-3, -5
13     test_array_7: .word 0
14
15 .text
16 tests:
17
18     test1:
19         la t6, test_array_1
20         li a1, 9
21         copy_array(t0,t6,a1)
22         jalr t2
23         print_str("Массив, переданный в программу: ")
24         print_array(t6,a1)
25         print_str("Получившийся массив B:")
26         print_array(t1, a1)
27         newline
28         j test2
29
30     test2:
31         la t1, B_array
32         la t6, test_array_2
33         li a1, 8
34         copy_array(t0,t6,a1)
35         jalr t2
36         print_str("Массив, переданный в программу: ")
37         print_array(t6,a1)
38         print_str("Получившийся массив B:")
39         print_array(t1, a1)
40         newline
41         j test3

```

```

42 test3:
43     la t1, B_array
44     la t6, test_array_3
45     li a1, 9
46     copy_array(t0,t6,a1)
47     la t2 algorithm
48     jalr t2
49     print_str("Массив, переданный в программу: ")
50     print_array(t6,a1)
51     print_str("Получившийся массив B:")
52     print_array(t1, a1)
53     newline
54     j test4
55
56 test4:
57     la t1, B_array
58     la t6, test_array_4
59     li a1, 6
60     copy_array(t0,t6,a1)
61     la t2 algorithm
62     jalr t2
63     print_str("Массив, переданный в программу: ")
64     print_array(t6,a1)
65     print_str("Получившийся массив B:")
66     print_array(t1, a1)
67     newline
68     j test5
69
70 test5:
71     la t1, B_array
72     la t6, test_array_5
73     li a1, 5
74     copy_array(t0,t6,a1)
75     la t2 algorithm
76     jalr t2
77     print_str("Массив, переданный в программу: ")
78     print_array(t6,a1)
79     print_str("Получившийся массив B:")
80     print_array(t1, a1)
81     newline
82     j test6
83
84 test6:
85     la t1, B_array
86     la t6, test_array_6
87     li a1, 3
88     copy_array(t0,t6,a1)
89     la t2 algorithm
90     jalr t2
91     print_str("Массив, переданный в программу: ")
92     print_array(t6,a1)
93     print_str("Получившийся массив B:")
94     print_array(t1, a1)
95     newline
96     j test7
97
98 test7:
99     la t1, B_array
100    la t6, test_array_7
101    li a1, 1
102    copy_array(t0,t6,a1)
103    la t2 algorithm
104    jalr t2
105    print_str("Массив, переданный в программу: ")
106    print_array(t6,a1)
107    print_str("Получившийся массив B:")
108    print_array(t1, a1)
109    newline
110    exit

```

Загружаем тестовый массив в регистр t5
Размер тестового массива
Макро для копирования массива
Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A, t1 – адрес массив B)

Ввод массива.

Загружаем тестовый массив в регистр t5
Размер тестового массива
Макро для копирования массива
Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A, t1 – адрес массив B)

Ввод массива.

Загружаем тестовый массив в регистр t5
Размер тестового массива
Макро для копирования массива
Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A, t1 – адрес массив B)

Ввод массива.

Загружаем тестовый массив в регистр t5
Размер тестового массива
Макро для копирования массива
Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A, t1 – адрес массив B)

Ввод массива.

Загружаем тестовый массив в регистр t5
Размер тестового массива
Макро для копирования массива
Основной алгоритм заполнения массива B (параметры: a1 – размер массивов, a3 – итератор, t0 – адрес массива A, t1 – адрес массив B)

Ввод массива.

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
```

```

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
63
64 # Вывод поэлементно массива с адресом начала в "%x".
65 .macro print_array(%x, %size)
66     push(a2)
67     push(a3)
68     mv a3, zero
69
70 loop_print:
71     lw a2 (%x)
72     print_int(a2)
73     print_char(' ')
74     addi %x, %x, 4
75     addi a3, a3, 1
76     bgt %size, a3, loop_print
77     newline
78
79     pop(a3)
80     pop(a2)
81 .end_macro
82
83 # Копирование 1 массива %x в регистр %array_copy, size – размер, передаваемого массива.
84 .macro copy_array(%array_copy, %x,%size)
85
86 .text
87     push(ra)
88     push(t6)
89     mv t5, %x
90     mv a3, zero
91
92 loop:
93     lw t6, (t5)
94     sw t6, (%array_copy)
95     addi t5, t5, 4
96     addi %array_copy, %array_copy, 4
97     addi a3, a3, 1
98     bgt %size, a3, loop
99 end_copy:
100     mv a3, zero
101     pop(t6)
102     pop(ra)
103 .end_macro

```

```

# Обнуляем итератор
# Загрузить текущий элемент из исходного массива в t6
# Сохранить элемент в A_array
# Перейти к следующему элементу в исходном массиве
# Перейти к следующему элементу в A_array
# Итератор
# Повторяем цикл пока итератор меньше размера массива
# Обнуляем итератор

```