АВС: ИДЗ-1

Октябрь 2024

Гобец Иван. БПИ 237. Вариант 21

Отчет на оценку 6-7

Отчет сразу начинает с оценки на 6-7, а не на 4-5, так как, спросив у семинариста, он сказал, что можно так сделать.

- Далее буду предоставлять блоки по различным частям кода (для понятности, чтобы не было все подряд).
- Ещё я описывал, все строчки кода комментариями для более понятного пояснения в отчете.

Блок .data

```
.data
        A_array:                            .space 40               # Выделяем 40 байт под массив A
        B_array:                            .space 40               # Выделяем 40 байт под массив B
        n:                                  .word   0               # Переменная для хранения количества элементов массива
        min_index:                          .word   0               # Хранение индекса минимального элемента

        start_information:                  .asciz "Welcome. This work is done by Gobets Ivan group BPI-237. Option - 21"
        print_get_n:                        .asciz "Please enter the number of array elements from 1 to 10: "
        invalid_n_value:                    .asciz "Invalid input, try again"
        print_get_number_for_fill_array:    .asciz "Enter an integer: "
        print_sorted_B_array:               .asciz "Sorted B array: "
        new_line:                           .asciz "\n"
        print_space:                        .asciz " "
```

Блок с макрасами

- **read_int(%x)** - Читает целое число из ввода и сохраняет его в указанный регистр. Параметры: %x - регистр для хранения результата.
- **print_int(%x)** - Выводит целое число, хранящееся в указанном регистре. Параметры: %x - регистр, содержащий число для вывода.
- **print_string_adress(%x)** - Выводит строку по указанному адресу. Параметры: %x - регистр, содержащий адрес строки.
- **print_string(%x)** - Выводит строку по метке. Параметры: %x - метка, содержащая адрес строки.
- **end_program** - Завершает выполнение программы. Параметры: Нет.
- **print_array(%array_ptr, %count)** - Выводит массив целых чисел. Параметры: %array_ptr - указатель на массив, %count - количество элементов в массиве.

Постарался больше пояснений оставить в комментариях, чтобы было нагляднее и легече читать код.

```
# Макрос для чтения целого числа
.macro  read_int(%x)                    # Определение макроса read_int, принимающего один параметр %x (регистр для хранения результата).
        li, a7, 5                       # Загружаем код системного вызова 5 (read_int) в регистр a7 для ввода целого числа.
        ecall                           # Выполняем системный вызов.
        mv %x, a0                       # Копируем полученное значение из регистра a0 в указанный регистр %x.
.end_macro

# Макрос для вывода целого числа
.macro print_int(%x)                    # Определение макроса print_int, принимающего регистр %x (число для вывода).
        li, a7, 1                       # Загружаем код системного вызова 1 (print_int) в регистр a7 для вывода числа.
        mv a0, %x                       # Копируем значение из %x в регистр a0 для передачи в системный вызов.
        ecall                           # Выполняем системный вызов.
.end_macro

# Макрос для вывода строки по адресу
.macro print_string_adress(%x)          # Определение макроса print_string_adress, принимающего адрес строки %x.
        li, a7, 4                       # Загружаем код системного вызова 4 (print_string) в регистр a7 для вывода
        mv a0, %x                       # Копируем адрес строки из %x в регистр
        ecall                           # Выполняем системный вызов.
.end_macro

# Макрос для вывода строки
.macro print_string(%x)                 # Определение макроса print_string, принимающего метку %x (адрес строки).
        la, a0, %x                      # Загружаем адрес строки (метки) %x в регистр a0.
        li, a7, 4                       # Загружаем код системного вызова 4 (print_string) в регистр a7 для вывода строки.
        ecall                           # Выполняем системный вызов.
.end_macro

# Макрос для завершения программы
.macro end_program                      # Определение макроса end_program без параметров.
        li, a7, 10                      # Загружаем код системного вызова 10 (exit) в регистр a7 для завершения программы.
        ecall                           # Выполняем системный вызов.
.end_macro

# Макрос для вывода массива целых чисел
.macro print_array(%array_ptr, %count)  # Определение макроса print_array, принимающего указатель на массив и его размер.
    mv   a5, %array_ptr                 # Копируем указатель на массив в регистр a5.
    mv   a6, %count                     # Копируем количество элементов массива в регистр a6.

print_array_loop:                       # Метка для начала цикла.
    beqz a6, end_print                  # Если a6 (количество элементов) равно 0, переход к метке end_print.
    lw   a0, (a5)                       # Загружаем следующий элемент массива (4 байта) в регистр a0.
    print_int(a0)                       # Вызываем макрос print_int для вывода элемента массива.
    print_string(print_space)           # Вызываем макрос print_string для вывода пробела между числами.

    addi a5, a5, 4                      # Смещаем указатель a5 на следующий элемент массива.
    addi a6, a6, -1                     # Уменьшаем счетчик элементов на 1.
    j    print_array_loop               # Переход к началу цикла.

end_print:                              # Метка для завершения вывода массива.
.end_macro
```

Блок main

- Выводится приветствие(“Welcome. This work is done by Gobets Ivan group BPI-237. Option - 21”) и перенос строки(new_line) через макрос **print_string**
- Загружаются адреса переменной n и массива A в регистры **t0** и **t1** соответственно.
- Далее прлучаем и сохраняем в **t3** ввод пользователя (int от 1 до 10, в след блоке будет подробнее пояснено, как это работает).
- “Копируем” **t3** в **t4**, чтобы во время заполненииния массива A не изменить значение **t3**.
- Далее заполняем “массив” A элементами введеными пользователем.
- “Копируем” **t3** в **t4**, чтобы во время заполненииния массива B не изменить значение **t3**.
- Формируем отсортированный по возрастанию массив B из элементов массива A и выводим массив B внутри массив B.
- Завершаем программу через макрос end_program.

```
.text
main:
        la      t0,     n                                       # Загружаем адресс на n в t0
        la      t1,     A_array                                 # Загружаем адрес массива A в t1

        # Регистр a0: используется для хранения ввода пользователя (get_and_save_n)
        # Регистр t3: хранит корректное значение n
        jal     get_and_save_n                                  # Сохраняем введенное количество элементов в переменной n


        mv      t4,     t3                                      # Скопирию n в t4, чтобы во время заполнения не изменять исходное значение n
        # Регистр t1: адрес массива A
        # Регистр t4: количество элементов n
        jal     fill_A_array                                    # "Заполняю массив" элементами, введеными пользователем

        mv      t4,     t3                                      # Скопирию n в t4, чтобы во время сортировки не изменять исходное значение n
        la      t2,     B_array                                 # Загружаем адрес массива A в t2

        # Регистр t1: адрес массива A
        # Регистр t2: адрес массива B
        # Регистр t3: количество элементов n
        jal     sort_A_array_and_save_in_B

        # Завершаем программу
        end_program
```

Блок получения числа от пользователя

- Выводим в консоль (**print_get_n**)"Please enter the number of array elements from 1 to 10: “.
- Считываем ввод пользователя через макрос **read_int** и делаем **jump** в подпрограмму **check_n_for_correctness**.
- Смотрим, если значение введеное пользователем (a0) оказалось меньше 1 или больше 10, то делаем **jump** в подпрограмму **incorrect_n_value_back_to_get_n**, а если ввод корректен, то записываем введеное значение из **a0** в **t3** и загружаем n в память по адресу **t0**, и делаем **return** в main.
- Подпрограмма **incorrect_n_value_back_to_get_n** : сначала выводим в конмоль “Invalid input, try again” и новую строчку и прыгаем в **get_and_save_n** (т.е. ,вообщем, мы будем запрашивать и пользователя n до тех пор, пока он не введет корректное значение).

```
        # Ввод n и сохранение его в памяти
        get_and_save_n:
                print_string(print_get_n)                       # Выводим содержимое a0 через макрос print_string
                read_int(a0)                                    # Получаем от пользвателя n (int) через макрос (значение будет лежать в регистре a0)

                # Регист a0 : введедеемый инт пользователем
                j       check_n_for_correctness                 # "Прыгаем" в другую подпрограмму, чтобы проверить что пользователь ввел корректные данные (int от 1 до 10)

        check_n_for_correctness:                                # Проверям, корректность ввода пользователя
                ble,    a0, zero, incorrect_n_value_back_to_get_n       # Если введеное значение пользователя меньше или равно 1, то "прыгаем" в обработку (incorrect_n_value_back_to_get_n)
                li      t2, 10                                  # Загружаем в t1 10
                bgt     a0, t2, incorrect_n_value_back_to_get_n         # Если введеное значение пользователя больше 10, то  "прыгаем" в обработку (incorrect_n_value_back_to_get_n)

                mv      t3, a0                                  # Перемещаем int от пользователя в t3
                sw      t3, (t0)                                # Загрузка n в память на хранение

                ret                                             # Если данные прошли все проверки, то возвращаемся по адрессу

        incorrect_n_value_back_to_get_n:
                print_string(invalid_n_value)                   # Выводим "Invalid input, try again"
                print_string(new_line)                          # Выводим переход строки
                j       get_and_save_n                          # "Прыгаем" в (get_and_save_n) для повторого запроса данных
```

Блок заполнения массива A

- Вывод сообщения пользователю - в начале выводится сообщение с просьбой ввести число для заполнения массива (“Enter an integer: “)
- Считываем ввод пользователя через макрос **read_int** по адрессу **a0**.

- Запись числа в массив - число, введенное пользователем, записывается по адресу, хранящемуся в регистре t1.
- Обновление адреса и счетчика - адрес в регистре t1 увеличивается на размер слова (4 байта), чтобы указать на следующий элемент массива.
- Счетчик оставшихся элементов (t4) уменьшается на 1.
- Проверка завершения заполнения - если счетчик оставшихся элементов (t4) не равен нулю, функция повторно вызывается для заполнения следующего элемента массива:

```
118         fill_A_array:
119                 print_string(print_get_number_for_fill_array)
120                 read_int(a0)
121
122                 sw      a0 (t1)                         # Запись числа по адресу в t0
123                 addi    t1 t1 4                         # Увеличим адрес на размер слова в байтах
124                 addi    t4 t4 -1                        # Уменьшим количество оставшихся элементов на 1
125                 bnez    t4 fill_A_array                 # Если осталось больше 0 запускаем повторно
126
127                 la      t1, A_array                     # После действий с адрессами, вернем адресс массива A в исходное положение
128                 jalr    ra                              # Возвращаемся по адрессу
129
```

Формирование отсортированный массива B из элементов массива A

```
130         sort_A_array_and_save_in_B:
131                 li      a3, 2147483647                  # Инициализируем минимальное значение как MAX_INT
132                 la      a5, min_index                   # Адрес для хранения индекса минимума
133                 li      a4, 0                           # Текущий индекс элемента
134
135                 find_min_element:
136                         lw      t6, (t1)                # "Выгружаем" элемент с "массива"
137                         bge     t6, a3, next_element    # Если текущий элемент больше максимального переходим к next_element
138                         mv      a3, t6                  # Если элемент меньше, то обновляем минимум
139                         sw      a4, (a5)                # Записываем текущий индекс элемента по адресу
140
141                 next_element:
142                         addi    t1, t1, 4               # Переход к следующему элементу массива
143                         addi    a4, a4, 1               # Увеличение индекса
144                         blt     a4, t3, find_min_element # Если не все элементы проверены, продолжаем
145
146                 write_min_to_B:
147                         sw      a3, (t2)                # Запись минимума в массив B
148                         addi    t2, t2, 4               # Переход к следующей позиции в B
149
150                         la      t1, A_array             # Восстановление адреса начала массива A
151                         lw      a2, (a5)                # Чтение индекса минимального элемента
152                         slli    a2, a2, 2               # Умножение индекса на 4 (размер слова)
153                         add     t1, t1, a2              # Получение адреса минимального элемента
154                         li      a0, 2147483647          # 2^32 - 1
155                         sw      a0, (t1)                # Замена минимального элемента на MAX_INT
156
157                         la      t1, A_array             # Восстановление адреса начала массива A
158                         addi    t4, t4, -1              # Уменьшение счетчика оставшихся элементов
159                         bnez    t4, sort_A_array_and_save_in_B # Если элементы остались, повторяем сортировку
160
161                         la      t2, B_array             # Возвращаю адресс в t2, так как мы его изменяли во время заполнения массива B
162                         print_string(print_sorted_B_array) # Выводим строку:
163                         print_array(t2, t3)             # Использую макрос для вывода элементоа массива (t2 - адресс, t3 - кол-во элементов в массиве)
164                         jalr    ra                      # Возвращаемся в main
165
166
167
```

- Инициализация переменных - инициализируем минимальное значение как максимальное целое число (MAX_INT) в **a3** и загружаем адрес для хранения индекса минимума в **a5** и текущий индекс в **a4**.
- Поиск минимального - выгражаем элемент с массива A в **t6** и сравнимаем его с минимальным, если он оказался меньше минимального, то просто записываем его в **a3** и записываем текущий индекс по адрессу. И переходим к слудующему элементу (если бы у нас при сравнении, оказалось, что элемент, который мы выгрузили оказался больше минимального, то сразу переходим к следующему элементу **next_elemt**).

- **next_element** - обновляем адресс и счетчик, адресс в **t1** увеличиваем на 4 байта и увеличиваем индекс на 1 в **a4**. Если индекс элемента меньше, чем длина массива - n (**t3**), то повторно запускаем поиск минимального элемента сделать **jump** в **find_min_element**.

Write_min_to_B

После того, как все элементы будут проверены и найден минимальный элемент в массиве A, то:

- Запись минимального элемента в массив B - записываем найденный минимальный элемент в массив B по адрессу **t2** и увеличиваем адресс **t2** на 4 байта.
- Восстановление адреса - восстанавливаем адресс массива A в **t1**. Далее в **a2** считываем индекс минимального элемента в массиве A. Выполняет сдвиг влево значения в регистре a2 на 2 бита, чтобы получить смещение в байтах, так как каждый элемент массива занимает 4 байта (размер слова). Добавляем к адрессу массива A смещение, чтобы получить адресс элемента по индексу, где лежит текущий минимум. После загржаем MAX_INT в **a0** и записываем его в найденный минимум в массиве A.
- Восстановлением адреса массива A в **t1** и уменьшаем счетчик оставшихся элементов в **t4** на 1. И если не все не все элементы массива A пройдены (если у нас n в **t4** не равно 0), то повторно запускаем поиск минимального элемента (прыгаем в **sort_A_array_and_save_in_B**), а как все элементы а массиве B будет полностью заполнен, восстанавливаем адресс массива B в **t2**, выводим через макрос **print_string** (**print_sorted_B_array**: “Sorted B array: “), через макрос **print_array** выводим массив B и делаем **return** в main.

Еще раз объясню как происходит сортировка массива. Мы ищем минимальный элемент в массиве A и записываем его в массив B. После этого в массиве A на место минимального элемента записываем максимальное целое число (MAX_INT). И так повторяем, пока не заполним массив B.

Тестовое пыкрытие

- Ввод числа n от пользователя и проверка его корректности. Если, число не в диапозоне от 1 до 10 включительно, программа запрашивает n снова.

```
Please enter the number of array elements from 1 to 10: -10
Invalid input, try again
Please enter the number of array elements from 1 to 10: 0
Invalid input, try again
Please enter the number of array elements from 1 to 10: 11
Invalid input, try again
Please enter the number of array elements from 1 to 10: 100
Invalid input, try again
Please enter the number of array elements from 1 to 10: 5
Enter an integer:
```

- Массив с одним элементом. Автоматически отсортирован.

```
Please enter the number of array elements from 1 to 10: 1
Enter an integer: 5
Sorted B array: 5
```

- Массив с одинаковыми элементами. Отсортирован корректно.

```
Please enter the number of array elements from 1 to 10: 3
Enter an integer: 123
Enter an integer: 123
Enter an integer: 123
Sorted B array: 123 123 123
```

- Массив с одинаковыми и разными элементами. Отсортирован корректно.

```
Please enter the number of array elements from 1 to 10: 5
Enter an integer: 123
Enter an integer: 123
Enter an integer: 123
Enter an integer: 8
Enter an integer: 10001
Sorted B array: 8 123 123 123 10001
```

- Массив с уже отсортированными элементами. Отсортирован корректно.

```
Please enter the number of array elements from 1 to 10: 5
Enter an integer: 10
Enter an integer: 20
Enter an integer: 30
Enter an integer: 40
Enter an integer: 50
Sorted B array: 10 20 30 40 50
```

- Массив с элементами в обратном порядке. Отсортирован корректно.

```
Please enter the number of array elements from 1 to 10: 5
Enter an integer: 50
Enter an integer: 40
Enter an integer: 30
Enter an integer: 20
Enter an integer: 1
Sorted B array: 1 20 30 40 50
```

- Массив с отрицательными элементами. Отсортирован корректно.

```
Please enter the number of array elements from 1 to 10: 5
Enter an integer: -123
Enter an integer: -10
Enter an integer: -1
Enter an integer: -4
Enter an integer: -1000
Sorted B array: -1000 -123 -10 -4 -1
```

- Массив с положительными и отрицательными элементами. Отсортирован корректно.

```
Please enter the number of array elements from 1 to 10: 8
Enter an integer: 52
Enter an integer: 123
Enter an integer: -123
Enter an integer: -100
Enter an integer: -1
Enter an integer: 0
Enter an integer: 65
Enter an integer: -50
Sorted B array: -123 -100 -50 -1 0 52 65 123
```

Отчет на 8

Реализация тестов.

Определяем 7 “массивов” данных для автоматического тестирования программы и строку tested_A_array для вывода (“Tested array: “)

SingleElementArray	IdenticalElementsArray	MixedElementsArray	AlreadySortedArray
Массив с одним элементом	Массив с одинаковыми элементами	Массив с одинаковыми и разными элементами	Уже отсортированный массив

ReverseOrderArray	NegativeElementsArray	MixedPositiveNegativeArray
Массив с элементами расположенными в обратном порядке	Массив с отрицательными элементами	Массив с положительными и отрицательными элементами

- Когда происходит автоматическое тестирование - если пользователь укажет длину масса равную 0, то начнется автотестирование.

- Было изменен текст для для пользователя. Теперт прописано, чтобы начать автотестирование нужно ввести 0.

```
8           print_get_n:                        .asciz "Please enter the number of array elements from 1 to 10 (autotest - 0): "
```

- Теперь, если пользователь ввел 0 (тут мы проверям через beq, zero, a0 ...), то мы делаем **jump** в подпрограмму **tests**.

```
103         # Ввод n и сохранение его в памяти
104         get_and_save_n:
105                 print_string(print_get_n)                       # Выводим содержимое a0 через макрос print_string
106                 read_int(a0)                                    # Получаем от пользвателя n (int) через макрос (значение будет лежать в регистре a0)
107
108                 beq, zero, a0, tests                            # Если пользователь ввел 0, то выполням авто-тестирование
109
110                 # Регист a0 : введедеый инт пользователем
111                 j       check_n_for_correctness                 # "Прыгаем" в другую подпрограмму, чтобы проверить что пользователь ввел корректные данные (int от 1 до 10)
```

- Есть 7 блоков тестирования (не везде написанны, так как код идентичен, только меняется длинна массива и его элементы).
- Сначала мы в **t1** записывем адресс на начала тестовых данных.
- В **t4** храним длину массива
- Далее нам нужно скопировать элемента тестового массива в массив A
- После восстанавливаем все значения, так как во время копирвоания массива они изменяются.
- После делаем **jal** в подпрограмму **run_test_case**

```
179         tests:                                                  # Тестовые случаи сортировки разных массивов
180                 test_1:                                         # Тест на массиве с одним элементом
181                         la t1, SingleElementArray               # Загрузить адрес массива с одним элементом в t1
182                         li t4, 1                                # Установить длину массива
183                         # t1 - adress array, t4 - len
184                         jal copy_array_to_A_array               # Вызов подпрограммы для копирования массива в A_array
185
186                         la t1, SingleElementArray               # Снова загрузить адрес массива в t1 (для вывода)
187                         li t4, 1                                # Заново устанавливаем длину массива (t4), так как мы ее изменяли во время копирования
188                         li t3, 1                                # Заново устанавливаем длину массива (t4), так как мы ее изменяли во время копирования
189
190                         # t1 - adress array, t4, t3 - len
191                         jal run_test_case                       # Запуск теста с этим массивом
192
193                 test_2:                                         # Массив с одинаковыми элементами
194                         la t1, IdenticalElementsArray
195                         li t4, 5
196                         # t1 - adress array, t4 - len
197                         jal copy_array_to_A_array
198
199                         la t1, IdenticalElementsArray
200                         li t4, 5
201                         li t3, 5
202
203                         # t1 -adress array, t4, t3 - len
204                         jal run_test_case
205
206                 test_3:                                         # Массив с одинаковыми и разными элементами
207                         la t1, MixedElementsArray
208                         li t4, 5
209                         # t1 - adress array, t4 - len
210                         jal copy_array_to_A_array
211
212                         la t1, MixedElementsArray
213                         li t4, 5
214                         li t3, 5
215
216                         # t1 - adress array, t4, t3 - len
217                         jal run_test_case
218
219                 test_4:                                         # Массив с одинаковыми и разными элементами
220                         la t1, AlreadySortedArray
221                         li t4, 5
222                         # t1 - adress array, t4 - len
223                         jal copy_array_to_A_array
224
225                         la t1, AlreadySortedArray
226                         li t4, 5
227                         li t3, 5
228
229                         # t1 - adress array, t4, t3 - len
230                         jal run_test_case
```

```
232                 test_5:                                             # Уже отсортированный массив
233                         la t1, ReverseOrderArray
234                         li t4, 5
235                         # t1 - adress array, t4 - len
236                         jal copy_array_to_A_array
237
238                         la t1, ReverseOrderArray
239                         li t4, 5
240                         li t3, 5
241
242                         # t1 - adress array, t4, t3 - len
243                         jal run_test_case
244
245                 test_6:                                             # Массив в обратном порядке
246                         la t1, NegativeElementsArray
247                         li t4, 5
248                         # t1 - adress array, t4 - len
249                         jal copy_array_to_A_array
250
251                         la t1, NegativeElementsArray
252                         li t4, 5
253                         li t3, 5
254
255                         # t1 - adress array, t4, t3 - len
256                         jal run_test_case
257
258                 test_7:                                             # Массив с положительными и отрицательными элементами
259                         la t1, MixedPositiveNegativeArray
260                         li t4, 5
261                         # t1 - adress array, t4 - len
262                         jal copy_array_to_A_array
263
264                         la t1, MixedPositiveNegativeArray
265                         li t4, 5
266                         li t3, 5
267
268                         # t1 - adress array, t4, t3 - len
269                         jal run_test_case
270
271                 end_program()
```

Копирование тестового массива в A

- В **t2** храним адресс массива A
- Далле попадаем в **copy_loop**, где смотрм если длина массива равна нулю, то переходим в подпрограмму **end_copy** где возвращаемся по адрессу.
- Выгружаем элемент и записываем его в массив A. Потом сдвигаем адресса массивов A и тестового на 4 байта и вычитаем из счетчика 1 в **t4** потом повторяем снова, пока не скопируем полностью.
- Главная подпрограмма тестирвоания **run_test_case** - сначала выводим две новых строки с помощью макроса **print_string** для красивого разделения блоков теста во время вывода.
- Загружаем в **t2** адресс на массив B.
- Далее выводим строку “Tested array: “ с помощью макроса **print_string** и выводим тестовый массив с помощью макроса **print_array**.
- Далее делаем **jump** в подпрограмму сортировки. **t1** адресс массива A, **t2** адрес масса B, **t3, t4** - длинна массива.

```
273         copy_array_to_A_array:                          # Подпрограмма для копирования массива в A_array
274                 la t2, A_array                          # Загрузить адрес A_array в t2
275
276                 copy_loop:
277                         beqz t4, end_copy               # Если длина массива (t4) равна 0, выйти из цикла
278                         lw t6, (t1)                     # Загрузить текущий элемент из исходного массива в t6
279                         sw t6, (t2)                     # Сохранить элемент в A_array
280                         addi t1, t1, 4                  # Перейти к следующему элементу в исходном массиве
281                         addi t2, t2, 4                  # Перейти к следующему элементу в A_array
282                         addi t4, t4, -1                 # Уменьшить счетчик длины массива
283                         j copy_loop                     # Повторить цикл
284                 end_copy:
285                         ret                             # Возврат из подпрограммы
286
287         run_test_case:                                  # Подпрограмма для выполнения теста
288                 print_string(new_line)                  # Печать новой строки для разделения выводов
289                 print_string(new_line)                  # Печать новой строки для разделения выводов
290
291                 la t2, B_array                          # Загрузить адрес массива B_array
292
293                 print_string(tested_A_array)            # Вывести текст "Tested array: "
294                 print_array(t1, t3)                     # Вывести содержимое тестируемого массива
295                 print_string(new_line)                  # Печать новой строки
296
297                 j sort_A_array_and_save_in_B            # Вызов сортировки массива A и сохранения результата в B_array
298
```

- Пример автоматического тестирования:

```
Welcome. This work is done by Gobets Ivan group BPI-237. Option - 21
Please enter the number of array elements from 1 to 10 (autotest - 0): 0


Tested array: 7
Sorted B array: 7

Tested array: 3 3 3 3 3
Sorted B array: 3 3 3 3 3

Tested array: 3 7 3 2 9
Sorted B array: 2 3 3 7 9

Tested array: 1 2 3 4 5
Sorted B array: 1 2 3 4 5

Tested array: 9 8 7 6 5
Sorted B array: 5 6 7 8 9

Tested array: -5 -9 -1 -3 -10
Sorted B array: -10 -9 -5 -3 -1

Tested array: -3 5 -7 2 0
Sorted B array: -7 -3 0 2 5
-- program is finished running (0) --
```

Отчет на 9

Макросы

- **read_int(%x)** - Читает целое число из ввода и сохраняет его в указанный регистр. Параметры: %x - регистр для хранения результата.

- **print_int(%x)** - Выводит целое число, хранящееся в указанном регистре. Параметры: %x - регистр, содержащий число для вывода.
- **print_string_adress(%x)** - Выводит строку по указанному адресу. Параметры: %x - регистр, содержащий адрес строки.
- **print_string(%x)** - Выводит строку по метке. Параметры: %x - метка, содержащая адрес строки.
- **end_program** - Завершает выполнение программы. Параметры: Нет.
- **print_array(%array_ptr, %count)** - Выводит массив целых чисел. Параметры: %array_ptr - указатель на массив, %count - количество элементов в массиве.

Постарался больше пояснений оставить в комментариях, чтобы было нагляднее и легече читать код.

```
24  # Макрос для чтения целого числа
25  .macro  read_int(%x)                    # Определение макроса read_int, принимающего один параметр %x (регистр для хранения результата).
26          li, a7, 5                       # Загружаем код системного вызова 5 (read_int) в регистр a7 для ввода целого числа.
27          ecall                           # Выполняем системный вызов.
28          mv %x, a0                       # Копируем полученное значение из регистра a0 в указанный регистр %x.
29  .end_macro
30
31  # Макрос для вывода целого числа
32  .macro print_int(%x)                    # Определение макроса print_int, принимающего регистр %x (число для вывода).
33          li, a7, 1                       # Загружаем код системного вызова 1 (print_int) в регистр a7 для вывода числа.
34          mv a0, %x                       # Копируем значение из %x в регистр a0 для передачи в системный вызов.
35          ecall                           # Выполняем системный вызов.
36  .end_macro
37
38  # Макрос для вывода строки по адресу
39  .macro print_string_adress(%x)          # Определение макроса print_string_adress, принимающего адрес строки %x.
40          li, a7, 4                       # Загружаем код системного вызова 4 (print_string) в регистр a7 для вывода
41          mv a0, %x                       # Копируем адрес строки из %x в регистр
42          ecall                           # Выполняем системный вызов.
43  .end_macro
44
45  # Макрос для вывода строки
46  .macro print_string(%x)                 # Определение макроса print_string, принимающего метку %x (адрес строки).
47          la, a0, %x                      # Загружаем адрес строки (метки) %x в регистр a0.
48          li, a7, 4                       # Загружаем код системного вызова 4 (print_string) в регистр a7 для вывода строки.
49          ecall                           # Выполняем системный вызов.
50  .end_macro
51
52  # Макрос для завершения программы
53  .macro end_program                      # Определение макроса end_program без параметров.
54          li, a7, 10                      # Загружаем код системного вызова 10 (exit) в регистр a7 для завершения программы.
55          ecall                           # Выполняем системный вызов.
56  .end_macro
57
58  # Макрос для вывода массива целых чисел
59  .macro print_array(%array_ptr, %count)  # Определение макроса print_array, принимающего указатель на массив и его размер.
60      mv   a5, %array_ptr                 # Копируем указатель на массив в регистр a5.
61      mv   a6, %count                     # Копируем количество элементов массива в регистр a6.
62
63  print_array_loop:                       # Метка для начала цикла.
64      beqz a6, end_print                  # Если a6 (количество элементов) равно 0, переход к метке end_print.
65      lw   a0, (a5)                       # Загружаем следующий элемент массива (4 байта) в регистр a0.
66      print_int(a0)                       # Вызываем макрос print_int для вывода элемента массива.
67      print_string(print_space)           # Вызываем макрос print_string для вывода пробела между числами.
68
69      addi a5, a5, 4                      # Смещаем указатель a5 на следующий элемент массива.
70      addi a6, a6, -1                     # Уменьшаем счетчик элементов на 1.
71      j    print_array_loop               # Переход к началу цикла.
72
73  end_print:                              # Метка для завершения вывода массива.
74  .end_macro
```

Отчет на 10

Разбиение программ по файлам:

- **main.asm** - главный файл программы, который содержит точку входа и вызов всех подпрограмм.

- **iomod** - файл, который содержит подпрограммы по вводу данных.
- **tests.asm** - файл, который содержит подпрограммы для автоматического тестирования программы.
- **generate_B** - файл, который содержит подпрограммы для формирования массива B из массива A.

Макросы выдедены в отдельную автономную библиотеку в файле **macrolib.asm**.