Самилык Анастасия ДЗ №3

Отчет (код программы с комментариями + примеры работы программы на крайних тестах)

```
.data
        len: .asciz "Количество элементов в массиве: "
        number_error: .asciz "Ошибка: количество элементов должно быть в диапазоне от 0 до 10"
        arg: .asciz "Введите число: "
        sum: .asciz "Сумма элементов массива: "
        ln:  .asciz "\n"
        odd: .asciz "Количество нечетных элементов массива: "
        even: .asciz "Количество четных элементов массива: "
        overflow: .asciz "Произошло переполнение!\n"
        correct_sum: .asciz "Сумма до переполнения: "
        number: .asciz "Количество просуммированных элементов: "

.align 2
        array: .space 64
        arrend:
.text
        la a0, len                          # Помещаем строку len в регистр a0
        li a7, 4                            # Системный вызов №4 - вывести null-terminated string
        ecall
        li a7, 5                            # Системный вызов №5 - чтение целого числа из консоли
        ecall
        mv s0, a0                           # Перемещение значения из регистра a0 в регистр s0
        blt  s0, zero, uncorrect_number     # Если число, введенное пользователем, меньше нуля - переход к uncorrect_number
        addi, s10, s10, 10                  # Помещаем в регистр s10 число 10
        bgt s0, s10, uncorrect_number       # Если число, введенное пользователем, больше десяти - переход к uncorrect_number
        j correct_number                    # Переход к uncorrect_number

uncorrect_number:
        li a7, 4                            # Системный вызов №4 - вывести null-terminated string
        la a0, number_error                 # Помещаем строку number_error в регистр a0
        ecall                               # Выводим текст ошибки
        li a0, 0                            # exit code
        li a7, 10                           # syscall exit
        ecall

```

```
correct_number:
        la t0 array                             # Счетчик
        la s1 arrend
        while:  la a0, arg                      # Помещаем в регистр a0 строку arg
                li a7, 4                        # Системный вызов №4 - вывести null-terminated string
                ecall
                li a7, 5                        # Системный вызов №5 - чтение целого числа из консоли
                ecall
                sw a0 (t0)                      # Запись введенного числа по адресу t0
                addi t0 t0 4                    # Увеличение адреса на размер слова в байтах

                addi s2, s2, 1                  # Увеличение счетчика, отвечающего за количество введенных элементов
                blt s2, s0, while               # Если счетчик меньше числа элементов, запускаем тело цикла еще раз
        la t0 array
        mv s2, zero

        sum_array:
                lw a0 (t0)                      # Загружаем в a0 значение по адресу t0
                mv s8, s3                       # Запишем в регистр s8 промежуточное значение суммы
                add s3, s3, a0                  # Добавим к счетчику суммы текущий элемент
                blt s8, zero, other             # Проверки на переполнение
                bgt s3, zero, ok
                blt a0, zero, ok
                j error                         # Если сумма была больше нуля, стала меньше нуля, а текущий элемент > 0 => произошло переполнение
                other:  blt s3, zero, ok
                        bgt a0, zero, ok        # Если сумма была меньше нуля, стала больше нуля, а текущий элемент < 0 => произошло переполнение
                error:  la a0, overflow         # Помещаем в регистр a0 строку overflow
                        li a7, 4                # Системный вызов №4 - вывести null-terminated string
                        ecall
                        la a0, correct_sum      # Помещаем в регистр a0 строку correct_sum
                        ecall
                        mv a0, s8               # Поместим в регистр a0 значение промежуточной суммы
                        li a7 1                 # Системный вызов №1 - вывести целое число
                        ecall
                        la a0, ln               # Перевод строки
                        li a7, 4                # Системный вызов №4 - вывести null-terminated string
                        ecall
                        la a0, number           # Помещаем в регистр a0 строку number
                        ecall
                        mv a0, s2               # Поместим в регистр a0 значение количества просуммированных элементов
                        li a7 1                 # Системный вызов №1 - вывести целое число
                        ecall
                        j end                   # Перейти к end
```

```
79
80              ok:
81              addi t0, t0, 4              # Увеличение адреса на размер слова в байтах
82              addi s2, s2, 1              # Увеличение счетчика, отвечающего за количество введенных элементов
83              blt s2, s0, sum_array       # Если счетчик меньше числа элементов, запускаем тело цикла еще раз
84
85
86      la a0, sum                          # Помещаем в регистр a0 строку sum
87      li a7, 4                            # Системный вызов №4 – вывести null-terminated string
88      ecall
89      mv a0, s3                           # Поместим в регистр a0 значение суммы
90      li a7 1                             # Системный вызов №1 – вывести целое число
91      ecall
92
93      end:
94      la a0, ln                           # Перевод строки
95      li a7, 4                            # Системный вызов №4
96      ecall
97
98      la t0 array                         # Загружаем в t0 array
99      mv s2, zero                         # Обнуляем регистр s2
100     addi s4, s4, 2                      # Помещаем в регист s4 цифру 2 для вычисления остатка в будущем
101     odd_even:
102             lw a0 (t0)                  # Загружаем в a0 значение по адресу t0
103             rem s5, a0, s4              # Загружаем в s5 значение остатка при делении элемента на 2
104             beq s5, zero, add_even      # Проверка на равество остатка нулю
105             addi s6, s6, 1              # Увеличиваем счетчик нечетных чисел, если остаток равен нулю
106             j then                      # Переход к then
107             add_even: addi s7, s7, 1    # Увеличиваем счетчик четных чисел, если остаток не равен нулю
108             then: addi t0, t0, 4        # Увеличим адрес на размер слова в байтах
109             addi s2, s2, 1              # Увеличение счетчика, отвечающего за количество пройденных элементов
110             blt s2, s0, odd_even        # Если счетчик меньше числа элементов, запускаем тело цикла еще раз
111
112
113     la a0, even                         # Помещаем в регистр a0 строку even
114     li a7, 4                            # Системный вызов №4 – вывести null-terminated string
115     ecall
116     mv a0, s7                           # Поместим в регистр a0 значение количества четных элементов массива
117     li a7 1                             # Системный вызов №1 – вывести целое число
118     ecall
119     la a0, ln                           # Перевод строки
120     li a7, 4                            # Системный вызов №4 – вывести null-terminated string
121     ecall
122
```

```
123         la a0, odd                          # Помещаем в регистр a0 строку odd
124         li a7, 4                            # Системный вызов №4 - вывести null-terminated string
125         ecall
126         mv a0, s6                           # Поместим в регистр a0 значение количества нечетных элементов массива
127         li a7 1                             # Системный вызов №1 - вывести целое число
128         ecall
129
130
131         li a7 10                            # Системный вызов №10 — остановка программы
132         ecall
133
```

Примеры скриншотов консоли, демонстрирующие тесты на переход через ноль, некорректное число элементов в массиве и переполнение.

Переход через 0:

```
Количество элементов в массиве: 5
Введите число: 1
Введите число: -3
Введите число: 4
Введите число: -4
Введите число: 2
Сумма элементов массива: 0
Количество четных элементов массива: 3
Количество нечетных элементов массива: 2
-- program is finished running (0) --
```

Некорректное количество элементов в массиве:

```
Количество элементов в массиве: -1
Ошибка: количество элементов должно быть в диапазоне от 0 до 10
-- program is finished running (0) --

Количество элементов в массиве: 11
Ошибка: количество элементов должно быть в диапазоне от 0 до 10
-- program is finished running (0) --
```

Переполнение при прибавлении положительного значения:

```
Количество элементов в массиве: 3
Введите число: 1999999999
Введите число: 999999999
Введите число: 1000000000
Произошло переполнение!
Сумма до переполнения: 1999999999
Количество просуммированных элементов: 1
Количество четных элементов массива: 1
Количество нечетных элементов массива: 2
-- program is finished running (0) --
```

Переполнение при прибавлении отрицательного значения:

```
Количество элементов в массиве: 3
Введите число: -2000000000
Введите число: -4
Введите число: -1000000000
Произошло переполнение!
Сумма до переполнения: -2000000004
Количество просуммированных элементов: 2
Количество четных элементов массива: 3
Количество нечетных элементов массива: 0
-- program is finished running (0) --
```