# Самилык Анастасия ДЗ №5 Отчет (программа на 8 + программа на 10)

Программа на оценку 8.

### <u>Файл main с вызовом подпрограмм.</u>

```
cin.asm
                               cout_cor.asm
                                                             macrolib.asm
 main.asm*
                     sum.asm
                                              cout_uncor.asm
    include "cin.asm"
    include "sum.asm"
   include "cout cor.asm"
    include "cout_uncor.asm"
    global main
6
7
8
    ain:
9
          jal cin
                              # Вызов подпрограммы для ввода данных
          jal sum
                               # Вызов подпрограммы для подсчета суммы
10
11
          beq s11 zero corr # Если переполнения не произошло, вызывается подпрограмма для случая без переполнения
12
          jal cout_uncor
                                 # Вызов подпрограммы для случая переполнения
          corr: jal cout cor # Вызов подпрограммы для случая без переполнения
13
14
15
16
```

#### <u>Файл cin с вводом данных и вызовом макроопределения read\_int из библиотеки с семинара</u>

```
main.asm
            cin.asm
                     sum.asm
                               cout_cor.asm
                                             cout uncor.asm
                                                            macrolib.asm
 1 cin:
    .include "macrolib.asm"
           len: .asciz "Количество элементов в массиве: "
           number error: .asciz "Ошибка: количество элементов должно быть в диапазоне от 0 до 10"
           arg: .asciz "Введите число: "
    .align 2
           array: .space 64
           arrend:
10
11
    .text
12
           la aO, len
                                                # Помещаем строку len в регистр a0
           li a7, 4
                                                # Системный вызов №4 - вывести null-terminated string
13
           ecall
14
15
           read int(s0)
                                                # Чтение целого числа
           blt s0, zero, uncorrect number
                                                # Если число, введеннное пользователем, меньше нуля - переход к uncorrect number
16
17
           addi, s10, s10, 10
                                                # Помещаем в регистр s10 число 10
18
           bgt s0, s10, uncorrect number
                                                # Если число, введеннное пользователем, больше десяти - переход к uncorrect number
19
           j correct number
                                                # Переход к uncorrect number
20
21
    uncorrect number:
           li a7, 4
                                                # Системный вызов №4 - вывести null-terminated string
22
23
           la a0, number error
                                                # Помещаем строку number error в регистр a0
                                                # Выводим текст ошибки
24
           ecall
           li a0, 0
2.5
                                                # exit code
26
           li a7, 10
                                                # syscall exit
27
           ecall
     correct number:
            la t0 array
                                                       # Счетчик
30
31
             la s1 arrend
32
             while: la a0, arg
                                                       # Помещаем в регистр a0 строку arg
33
                     li a7, 4
                                                       # Системный вызов №4 - вывести null-terminated string
34
                     ecall
35
                                                       # Чтение целого числа
                     read int(a0)
                     sw a0 (t0)
                                                       # Запись введенного числа по адресу t0
36
                     addi t0 t0 4
37
                                                       # Увеличение адреса на размер слова в байтах
38
39
                     addi s2, s2, 1
                                                       # Увеличение счетчика, отвечающего за количество введенных элементов
40
                     blt s2, s0, while
                                                      # Если счетчик меньше числа элементов, запускаем тело цикла еще раз
41
             la t0 array
             mv s2, zero
42
43
44
```

#### <u>Файл sum с суммирование элементов</u>

```
main.asm
            cin.asm
                     sum.asm
                                cout cor.asm
                                               cout uncor.asm
                                                               macrolib.asm
1
    sum:
    .data
3
           odd: .asciz "Количество нечетных элементов массива: "
           even: .asciz "Количество четных элементов массива: "
 4
 5
     .text
           sum array:
 6
                   lw a0 (t0)
                                                  # Загружаем в a0 значение по адресу t0
7
                   mv s8, s3
                                                  # Запишем в регистр s8 промежуточное значение суммы
8
9
                   add s3, s3, a0
                                                  # Добавим к счетчику суммы текущий элемент
                   blt s8, zero, other
                                                  # Проверки на переполнение
10
11
                   bgt s3, zero, ok
12
                   blt a0, zero, ok
13
                   j error
                                                  # Если сумма была больше нуля, стала меньше нуля, а текуший элемент > 0 => произошло переполнение
14
                   other: blt s3, zero, ok
15
                           bgt a0, zero, ok
                                                  # Если сумма была меньше нуля, стала больше нуля, а текуший элемент < 0 => произошло переполнение
                                                  # Положим в регистр s11 число 1, чтобы позднее понять, что была ошибка
16
                   error: addi s11, s11, 1
17
                           ret
18
19
                   ok:
                   addi t0, t0, 4
                                                  # Увеличение адреса на размер слова в байтах
20
21
                   addi s2, s2, 1
                                                  # Увеличение счетчика, отвечающего за количество введенных элементов
                   blt s2, s0, sum_array
22
                                                  # Если счетчик меньше числа элементов, запускаем тело цикла еще раз
23
24
   ret
25
```

<u>Файлы с выводом элементов для случая с переполнением и без + использование макроопределения read\_int из библиотеки с семинара</u>

```
.macro print int (%x)
          li a7, 1
          mv a0, %x
 3
           ecall
   .end macro
   cout cor:
    .data
           ans sum: .asciz "Сумма элементов массива: "
 9
           numb el: .asciz "\nКоличество просуммированных элементов: "
10
    .text
11
12
                                                 # Помещаем в регистр a0 строку ans sum
13
           la a0, ans sum
                                                 # Системный вызов №4 - вывести null-terminated string
           li a7, 4
14
           ecall
15
           print int (s3)
                                                 # Вывод целого числа из региистра s3
16
17
           la a0, numb el
                                                 # Помещаем в регистр а0 строку numb el
18
           li a7, 4
                                                 # Системный вызов №4 - вывести null-terminated string
19
           ecall
20
21
                                                 # Вывод целого числа из регистра s2
           print int (s2)
22
23
           ecall
                                                 # Системный вызов №10 — остановка программы
           li a7 10
24
25
           ecall
```

```
.macro print int (%x)
          li a7, 1
         m∨ a0, %x
         ecall
   .end macro
  cout uncor:
7 .data
          overflow: .asciz "Произошло переполнение!"
          correct sum: .asciz "\nСумма до переполнения: "
           el number: .asciz "\nКоличество просуммированных элементов: "
10
11
    .text
          la aO, overflow # Помещаем в регистр aO строку overflow
12
          li a7, 4
                              # Системный вызов Nº4 - вывести null-terminated string
13
          ecall
14
                              # Помещаем в регистр a0 строку correct sum
15
          la aO, correct sum
          ecall
16
17
          print int (s8)
                              # Вывод целого числа из регистра s8
          la aO, el number # Помещаем в регистр aO строку el number
18
          li a7, 4
                                # Системный вызов №4 - вывести null-terminated string
19
           ecall
20
          print int (s2) # Вывод целого числа из регистра s2
21
22
          li a7 10
                       # Системный вызов №10 — остановка программы
23
           ecall
```

#### Программа на оценку 10.

```
В решении 2 файла: один - таіп, второй - библиотека макроопределений
            lib.asm
 main.asm
    .include "lib.asm"
    global main
 3
   main:
 6
           cin
                                   # Вызов макроопределения для ввода данных
           sum
                                   # Вызов макроопределения для подсчета суммы
                                   # Если переполнения не произошло, вызывается макроопределения для случая без переполнения
 8
           beg sll zero corr
           cout_uncor
                                   # Вызов макроопределения для случая переполнения
 9
           j exit code
10
11
           corr: cout_cor
                                   # Вызов макроопределения для случая без переполнения
           exit code: exit
12
13
14
```

## lib.asm - файл с такими макросами, как:

```
    cin - ввод данных в массив
    read_int(%x) - ввод целого числа
    sum - суммирование элементов
    print_int (%x) - вывод целого числа
    cout_cor - вывод для случая без переполнения
    cout_uncor - вывод для случая с переполнением
    exit - завершение программы
```

```
main.asm
           lib.asm
1
    .macro cin
 2
 3
                    len: .asciz "Количество элементов в массиве: "
                    number error: .asciz "Ошибка: количество элементов должно быть в диапазоне от О до 10"
                    arg: .asciz "Введите число: "
 5
 6
            .align 2
 8
                    array: .space 64
 9
                    arrend:
            .text
10
                    la aO, len
                                                            # Помещаем строку len в регистр а0
11
                                                            # Системный вызов №4 - вывести null-terminated string
12
                    li a7, 4
13
                    ecall
14
                    read int(s0)
                                                            # Чтение целого числа
15
                    blt s0, zero, uncorrect_number
                                                            # Если число, введеннное пользователем, меньше нуля - переход к uncorrect number
                    addi, s10, s10, 10
                                                            # Помещаем в регистр s10 число 10
16
                    bgt s0, s10, uncorrect number
                                                            # Если число, введенные пользователем, больше десяти - переход к uncorrect number
17
18
                    j correct number
                                                            # Переход к uncorrect number
19
            uncorrect number:
20
                    li a7, 4
                                                            # Системный вызов Nº4 - вывести null-terminated string
21
22
                    la aO, number error
                                                            # Помещаем строку number error в регистр a0
23
                    ecall
                                                            # Выволим текст ошибки
                    li a0, 0
                                                            # exit code
24
25
                    li a7, 10
                                                            # syscall exit
26
                    ecall
27
28
            correct number:
29
                    la tO array
                                                            # Счетчик
30
                    la sl arrend
                    while: la aO, arg
                                                            # Помещаем в регистр a0 строку arg
31
32
                            li a7, 4
                                                            # Системний визов №4 - вивести null-terminated string
33
                            ecall
34
                            read int(a0)
                                                            # Чтение целого числа
                            sw a0 (t0)
                                                            # Запись введенного числа по адресу t0
35
                            addi t0 t0 4
                                                            # Увеличение адреса на размер слова в байтах
36
37
38
                            addi s2, s2, 1
                                                            # Увеличение счетчика, отвечающего за количество введенных элементов
                            blt s2, s0, while
39
                                                            # Если счетчик меньше числа элементов, запускаем тело цикла еще раз
                    la t0 array
40
                    mv s2, zero
41
42
43
    .end macro
```

```
46 .macro read_int(%x)
47
       li a7, 5
       ecall
48
       mv %x, a0
49
    .end macro
50
51
   .macro sum
52
53
            .data
                    odd: .asciz "Количество нечетных элементов массива: "
54
                    even: .asciz "Количество четных элементов массива: "
55
56
            .text
57
                    sum array:
                           lw a0 (t0)
                                                           # Загружаем в а0 значение по адресу t0
58
                           mv s8, s3
                                                           # Запишем в регистр s8 промежуточное значение суммы
59
                                                           # Добавим к счетчику суммы текущий элемент
                           add s3, s3, a0
60
                                                           # Проверки на переполнение
61
                           blt s8, zero, other
                           bgt s3, zero, ok
62
                           blt aO, zero, ok
63
                                                           # Если сумма была больше нуля, стала меньше нуля, а текущий элемент > 0 => произошло переполнение
                           j error
64
                           other: blt s3, zero, ok
65
                                   bgt a0, zero, ok
                                                           # Если сумма была меньше нуля, стала больше нуля, а текущий элемент < 0 => произошло переполнение
66
67
                           error: addi sll, sll, 1
                                                           # Положим в регистр s11 число 1, чтобы позднее понять, что была ошибка
68
                                   j end
69
70
                           ok:
                           addi t0, t0, 4
                                                           # Увеличение адреса на размер слова в байтах
71
72
                           addi s2, s2, 1
                                                           # Увеличение счетчика, отвечающего за количество введенных элементов
                           blt s2, s0, sum array
                                                           # Если счетчик меньше числа элементов, запускаем тело цикла еще раз
73
                           end:
74
75
    .end macro
76
77
78
    .macro print_int ( %x)
79
           li a7, 1
80
           mv a0, %x
81
            ecall
82
83 .end macro
```

```
86
    .macro cout_cor
87
88
             .data
                     ans sum: .asciz "Сумма элементов массива: "
89
                     numb el: .asciz "\nКоличество просуммированных элементов: "
90
            .text
91
92
93
                     la a0, ans sum
                                                             # Помещаем в регистр а0 строку ans sum
                                                             # Системный вызов №4 — вывести null-terminated string
94
                     li a7, 4
                     ecall
95
                                                             # Вывод целого числа из региистра в3
96
                    print_int (s3)
97
98
                    la aO, numb_el
                                                             # Помещаем в регистр a0 строку numb el
                                                             # Системный вызов №4 - вывести null-terminated string
99
                    li a7, 4
                     ecall
100
L01
L02
                    print_int ( s2)
                                                             # Вывод целого числа из регистра s2
103
L04
    .end macro
L05
L06
    .macro cout_uncor
L07
             .data
108
                     overflow: .asciz "Произошло переполнение!"
L09
110
                     correct sum: .asciz "\nCymma до переполнения: "
111
                     el number: .asciz "\nКоличество просуммированных элементов: "
112
            .text
113
                    la aO, overflow
                                                             # Помещаем в регистр a0 строку overflow
                    li a7, 4
                                                             # Системный вызов №4 — вывести null-terminated string
114
                     ecall
115
116
                    la a0, correct_sum
                                                             # Помещаем в регистр a0 строку correct sum
117
                     ecall
                                                             # Вывод целого числа из регистра s8
118
                    print int ( s8)
                                                             # Помещаем в регистр a0 строку el number
119
                    la aO, el number
                                                             # Системный вызов №4 — вывести null-terminated string
120
                    li a7, 4
121
                     ecall
                    print_int (s2)
                                                             # Вывод целого числа из регистра s2
122
    .end macro
L23
L24
L25
    .macro exit
                    li a7 10
                                                             # Систелений вызов №10 — остановка программы
L26
                     ecall
L27
    .end macro
L28
```