ACS. Homework 4

Александр Васюков | БПИ235 (239)

О программе solution.asm

В секции .data выделено:

1. 4 байта на размер массива - метка n.
2. 40 байтов на массив (10 чисел) - метка array.
3. Текстовые сообщения для взаимодействия с пользователем.

Пользователю предлагается ввести размер массива от 1 до 10 включительно. Если введено некорректное число, выводится сообщение `Bro, you need input a number from 1 to 10. Try again!` и предлагается ввести размер снова.

Дальше происходит заполнение массива. Пользователю предлагается ввести i-й элемент, которые поочерёдно сохраняются в array.

Подготовка к суммированию. Загружаются в регистры:

1. `t0` - адрес начала массива (`array`).
2. `t1` - начальная сумма (`0`).
3. `t2` - счётчик элементов (`0`).
4. `t3` - размер массива (`n`).

Суммирование. В регистр `t4` загружается элемент массива и происходит проверка переполнения суммы, если сумма и слагаемое имеют одинаковый знак. Если знаки разные, то переполнения быть не может. Дальше происходит обновление суммы, счётчика и переход к следующему элементу.

Проверка на переполнение. Если новая сумма и прибавленный элемент имеют разные знаки, то произошло переполнение. Тогда программа перепрыгивает на метку `error_overflow`, иначе на метку `no_overflow`.

Если переполнение произошло (`error_overflow`), выводится сообщение `Overflow!`, а также последняя корректная сумма и количество посчитанных элементов.

Если сумма правильно посчиталась и переполнения не было, то выводится сумма элементов массива.

Завершается программа вызовом системной команды №10 (останов).

solution.asm

Github: https://github.com/vasyukov1/HSE-FCS-SE-2-year/blob/main/ACS/Homework/4/solution.asm

```
.data
n:              .word 0
array:          .space 40

input_size:     .asciz "Input the size of array from 1 to 10: "
input_item_p1:  .asciz "Input "
input_item_p2:  .asciz " element: "
result_prompt:  .asciz "\nSum: "
error_prompt:   .asciz "\nBro, you need input a number from 1 to 10. Try again!\n"
overflow_p1:    .asciz "\nOverflow!\nSum: "
overflow_p2:    .asciz "\nCount of elements: "


.text
size:   # Output prompt for input the size of array
        li a7 4
        la a0 input_size
        ecall

        # Input the size of array
        li a7 5
        ecall
        mv t0 a0

        # Check that the array's size is correct
        li t1 1
        blt t0 t1 error
```

```
        li t1 10
        bgt t0 t1 error

        # Load the array's size in memory
        la t1 n
        sw t0 (t1)

        # Load the element in memory
        la t0 array

        li t1 1
        lw t3 n

fill:   # Prompt for input the element of array
        li a7 4
        la a0 input_item_p1
        ecall

        li a7 1
        mv a0 t1
        ecall

        li a7 4
        la a0 input_item_p2
        ecall

        li a7 5
        ecall
        mv t2 a0

        # Save element and go to the next
        sw t2 (t0)
        addi t0 t0 4
        addi t1 t1 1
        ble t1 t3 fill

sum_preparation:
        la t0 array     # Start of array
        li t1 0         # Sum of array
        li t2 0         # Counter of elements
        lw t3 n         # Size of array

add_el: lw t4 (t0)      # Read element
        # Check overflow
        xor t5 t1 t4
        bgez t5 check_overflow
no_overflow:
        # Save sum
        add t1 t1 t4
        # Go to next element
        addi t0 t0 4
        addi t2 t2 1
        blt t2 t3 add_el
        j result

check_overflow:
        # If sum and term have different signs, overflow occurred
        add t6 t1 t4
        xor t5 t1 t6
        bltz t5 error_overflow
        j no_overflow

result:
        # Output result
        li a7 4
        la a0 result_prompt
        ecall

        li a7 1
        mv a0 t1
        ecall
        j end

error_overflow:
        # Message about overflow and result
        li a7 4
        la a0 overflow_p1
```

```
        ecall

        li a7 1
        mv a0 t1
        ecall

        li a7 4
        la a0 overflow_p2
        ecall

        li a7 1
        mv a0 t2
        ecall

        j end

error:
        # Message that the array's size is incorrect and jump to new attempt
        li a7 4
        la a0 error_prompt
        ecall
        j size

end:
        # Stop
        li a7 10
        ecall
```

Результат работы

```
Input the size of array from 1 to 10: 3
Input 1 element: 234
Input 2 element: 766
Input 3 element: 984

Sum: 1984
-- program is finished running (0) --

Input the size of array from 1 to 10: 123

Bro, you need input a number from 1 to 10. Try again!
Input the size of array from 1 to 10: -2

Bro, you need input a number from 1 to 10. Try again!
Input the size of array from 1 to 10: 5
Input 1 element: -2342
Input 2 element: 325
Input 3 element: 2354325
Input 4 element: -263234
Input 5 element: 888

Sum: 2089962
-- program is finished running (0) --

Input the size of array from 1 to 10: 4
Input 1 element: 2147483640
Input 2 element: 2
Input 3 element: 10
Input 4 element: -234

Overflow!
Sum: 2147483642
Count of elements: 2
-- program is finished running (0) --

Input the size of array from 1 to 10: 3
Input 1 element: -2147483647
Input 2 element: -123
Input 3 element: -234

Overflow!
Sum: -2147483647
Count of elements: 1
-- program is finished running (0) --
```

Модернизированная программа

Программа solution_with_parity.asm считает сумму элементов массива, а также количество чётных и нечётных элементов.

Программа работает аналогично solution.asm, за исключением некоторых изменений:

1. В секции `.data` добавлено по 4 байта для хранения количества чётных и нечётных элементов - метки `even` и `odd` соответственно. Добавлены текстовые сообщения для вывода информации о чётности и 4 байта - метка `flag_sum`, которая является флагом, было переполнение (`1`) или нет (`0`).

2. При суммировании, получая элемент, сначала проверяем его на чётность командой `andi t5 t4 1` - операция побитового `t4 and 1`. Если число `t4` чётное, то результат `t5` будет равен `0`, иначе `1`. Далее программа прыгает либо на метку `even_add`, где счётчик количества чётных чисел увеличивается, либо на аналогичную для нечётных метку `odd_add`.
3. Если произошло переполнение, меняется флаг `flag_sum` с `0` на `1`. Это необходимо, чтобы перепрыгивать через суммирование, если произошло переполнение, но количество чётных и нечётных чисел ещё недосчитано.
4. В конце, перед остановкой программы, выводится количество чётных и нечётных чисел.

P.S. можно было считать только количество чётных элементов, а количество нечётных получить вычетом из размера массива количества чётных.

solution_with_parity.asm

Github: https://github.com/vasyukov1/HSE-FCS-SE-2-year/blob/main/ACS/Homework/4/solution_with_parity.asm

```
.data
n:              .word 0
array:          .space 40
odd:            .word 0
even:           .word 0
flag_sum:       .word 0

input_size:     .asciz "Input the size of array from 1 to 10: "
input_item_p1:  .asciz "Input "
input_item_p2:  .asciz " element: "
result_prompt:  .asciz "\nSum: "
error_prompt:   .asciz "\nBro, you need input a number from 1 to 10. Try again!\n"
overflow_p1:    .asciz "\nOverflow!\nSum: "
overflow_p2:    .asciz "\nCount of elements: "
odd_prompt:     .asciz "\n\nOdd: "
even_prompt:    .asciz "\nEven: "


.text
size:   # Output prompt for input the size of array
        li a7 4
        la a0 input_size
        ecall

        # Input the size of array
        li a7 5
        ecall
        mv t0 a0

        # Check that the array's size is correct
        li t1 1
        blt t0 t1 error
        li t1 10
        bgt t0 t1 error

        # Load the array's size in memory
        la t1 n
        sw t0 (t1)

        # Load the element in memory
        la t0 array

        li t1 1
        lw t3 n

fill:   # Prompt for input the element of array
        li a7 4
        la a0 input_item_p1
        ecall

        li a7 1
        mv a0 t1
        ecall

        li a7 4
        la a0 input_item_p2
        ecall

        li a7 5
        ecall
```

```
        mv t2 a0

        # Save element and go to the next
        sw t2 (t0)
        addi t0 t0 4
        addi t1 t1 1
        ble t1 t3 fill

sum_preparation:
        la t0 array     # Start of array
        li t1 0         # Sum of array
        li t2 0         # Counter of elements
        lw t3 n         # Size of array

add_el: lw t4 (t0)      # Read element

        # Parity check
        andi t5 t4 1

        beqz t5 even_add
        j odd_add

after_parity:
        lw t6 flag_sum
        bnez t6 next_el
        # Check overflow
        xor t5 t1 t4
        bgez t5 check_overflow
no_overflow:
        # Save sum
        add t1 t1 t4
next_el:
        # Go to next element
        addi t0 t0 4
        addi t2 t2 1
        blt t2 t3 add_el
        j result

even_add:
        lw t5 even
        addi t5 t5 1
        la t6 even
        sw t5 (t6)
        j after_parity
odd_add:
        lw t5 odd
        addi t5 t5 1
        la t6 odd
        sw t5 (t6)
        j after_parity

check_overflow:
        # If sum and term have different signs, overflow occurred
        add t6 t1 t4
        xor t5 t1 t6
        bltz t5 error_overflow
        j no_overflow

result:
        # Output result, if sum is correct
        lw t6 flag_sum
        bnez t6 end

        li a7 4
        la a0 result_prompt
        ecall

        li a7 1
        mv a0 t1
        ecall
        j end

error_overflow:
        # Change sum's flag
        lw t5 flag_sum
        addi t5 t5 1
        la t6 flag_sum
```

```
        sw t5 (t6)
        # Message about overflow and result
        li a7 4
        la a0 overflow_p1
        ecall

        li a7 1
        mv a0 t1
        ecall

        li a7 4
        la a0 overflow_p2
        ecall

        li a7 1
        mv a0 t2
        ecall

        j next_el

error:  # Message that the array's size is incorrect and jump to new attempt
        li a7 4
        la a0 error_prompt
        ecall
        j size

end:    # Output count of odd elements
        li a7 4
        la a0 odd_prompt
        ecall

        li a7 1
        lw a0 odd
        ecall

        # Output count of even elements
        li a7 4
        la a0 even_prompt
        ecall

        li a7 1
        lw a0 even
        ecall

        # Stop
        li a7 10
        ecall
```

Результат работы

```
Input the size of array from 1 to 10: 6
Input 1 element: 12
Input 2 element: 13
Input 3 element: 13
Input 4 element: 25
Input 5 element: 66
Input 6 element: 7

Sum: 136

Odd: 4
Even: 2
-- program is finished running (0) --

Input the size of array from 1 to 10: 5
Input 1 element: 2147483640
Input 2 element: 2
Input 3 element: 2
Input 4 element: 77
Input 5 element: 9

Overflow!
Sum: 2147483644
Count of elements: 3

Odd: 2
Even: 3
-- program is finished running (0) --
```