# ИДЗ-1_Хадзакос_НА_236

## Вариант 39

Файлы с решениями для чтения в удобном формате и запуска можно скачать с диска: https://disk.yandex.ru/d/wU_1YsRdfQqV5A

### Условие

Сформировать массив B, элементы которого являются площадью квадратов со сторонами указанные в массиве A. Операцию умножения реализовать через сложение и/или сдвиги. При переполнении в массив B записывать нули.

### Код на 7 баллов

```
.data
prompt_size: .asciz "Введите размер массива: "
prompt_elem: .asciz "Введите элемент: "
result_msg: .asciz "Площади квадратов: "
array_size: .word 0
.align 2
array_A: .space 64 # Выделяем место для 16 элементов (16 * 4 байта)
array_B: .space 64 # Выделяем место для 16 элементов (16 * 4 байта)

.text
.global main

main:
    # Ввод размера массива
    la a0, prompt_size
    li a7, 4
    ecall
    li a7, 5
    ecall
    mv s0, a0  # s0 = размер массива
    sw s0, array_size, t0  # Сохраняем размер массива

    # Установка указателей на массивы A и B
    la s1, array_A  # s1 = указатель на массив A
    la s2, array_B  # s2 = указатель на массив B
```

```
    # Ввод элементов массива A
    mv t0, s1
    li t1, 0
input_loop:
    la a0, prompt_elem
    li a7, 4
    ecall
    li a7, 5
    ecall
    sw a0, (t0)
    addi t0, t0, 4
    addi t1, t1, 1
    blt t1, s0, input_loop

    # Вызов подпрограммы обработки массивов
    mv a0, s1  # адрес массива A
    mv a1, s2  # адрес массива B
    mv a2, s0  # размер массивов
    jal square_array

    # Вызов подпрограммы вывода массива
    mv a0, s2 # адрес массива B
    mv a1, s0 # размер массива
    jal print_array

    # Выход из программы
    li a7, 10
    ecall

# Подпрограмма обработки массивов
square_array:
    addi sp, sp, -8
    sw s3, 0(sp) # сохраняем на стеке
    sw s4, 4(sp)  # сохраняем на стеке

    mv t0, a0  # указатель на A
    mv t1, a1  # указатель на B
    mv t2, a2  # счетчик элементов
    li t3, 0   # индекс текущего элемента

process_loop:
    lw t4, (t0)  # загрузка элемента из A
```

```
    mv t5, t4     # копия для умножения
    li t6, 0      # результат умножения
    li s3, 32     # счетчик битов

multiply:
    andi s4, t5, 1
    beqz s4, shift
    add t6, t6, t4
    bltu t6, t4, overflow
    bltz t6, overflow

shift:
    slli t4, t4, 1
    srli t5, t5, 1
    addi s3, s3, -1
    bnez s3, multiply

    sw t6, (t1)  # сохранение результата в B
    j next

overflow:
    sw zero, (t1)

next:
    addi t0, t0, 4
    addi t1, t1, 4
    addi t3, t3, 1
    blt t3, t2, process_loop

   # Восстанавливаем используемые значения
    lw s3, 0(sp)
    lw s4, 4(sp)
    addi sp, sp, 8
    ret

# Подпрограмма вывода результата
print_array:
    mv t0, a0 # указатель на B
    mv t1, a1 # размер массива
    # Вывод результатов
    la a0, result_msg
    li a7, 4
    ecall
```

```
    li t2, 0
output_loop:
    lw a0, (t0)
    li a7, 1
    ecall
    li a0, ' '
    li a7, 11
    ecall
    addi t0, t0, 4
    addi t2, t2, 1
    blt t2, t1, output_loop

    ret
```

Используемые s-регистры сохраняются на стеке и восстанавливаются после выполнения подпрограммы.

## Результаты тестирования

### Тест 1

```
Введите размер массива: 5
Введите элемент: 1
Введите элемент: 2
Введите элемент: 3
Введите элемент: 4
Введите элемент: 5
Площади квадратов: 1 4 9 16 25
-- program is finished running (0) --
```

### Тест 2

```
Введите размер массива: 5
Введите элемент: 1000000
Введите элемент: 9
Введите элемент: 14
Введите элемент: 16
Введите элемент: -20
Площади квадратов: 0 81 196 256 0
-- program is finished running (0) --
```

Пояснение:

1. сторона квадрата не может быть равна -20, поэтому вывод должен равняться 0.
2. $1000000^2 > INT\_MAX$

### Тест 3

```
Введите размер массива: 10
Введите элемент: 993
Введите элемент: 12
Введите элемент: 11
Введите элемент: 20
Введите элемент: 15
Введите элемент: 41
Введите элемент: 62
Введите элемент: 57
Введите элемент: 101
Введите элемент: 13
Площади квадратов: 986049 144 121 400 225 1681 3844 3249 10201 169
-- program is finished running (0) --
```

## Программа на 8 баллов

```
# Тестовые случаи
.data
test_array_1: .word 2, 3, 5, 7, 9 # Тест 1
test_array_2: .word 1000000, 11, 29, 35, 12 # Тест 2
test_array_3: .word 33, 14, 2000000, -20, 0 # Тест 3
test_array_4: .word 0, 1, 4, 6, 8 # Тест 4
test_array_5: .word 111, 222, 333, 444, 993 # Тест 5
```

```
test_array_6: .word 1, 2, 3, 4, 5, 6, 7, 8, 9 # Тест 6(экстра)
result_array: .word 0, 0, 0, 0, 0 # Массив результата
result_array_extra: .word 0, 0, 0, 0, 0, 0, 0, 0, 0 # Массив
результата(для другого размера)
array_size: .word 5
array_size_extra: .word 9
test_msg: .asciz "Тест "
result_msg: .asciz "Площади квадратов: "
endl: .asciz "\n"

.text
.global test_main


test_main:
    la s0, array_size
    lw s0, (s0)

    # Тест 1
    la a0, test_msg
    li a7, 4
    ecall
    li a0, 1
    li a7, 1
    ecall
    la a0, endl
    li a7, 4
    ecall

    la a0, test_array_1
    la a1, result_array
    mv a2, s0
    jal square_array

    la a0, result_array
    mv a1, s0
    jal print_array

    # Тест 2
    la a0, test_msg
    li a7, 4
    ecall
    li a0, 2
```

```
    li a7, 1
    ecall
    la a0, endl
    li a7, 4
    ecall

    la a0, test_array_2
    la a1, result_array
    mv a2, s0
    jal square_array

    la a0, result_array
    mv a1, s0
    jal print_array

    # Тест 3
    la a0, test_msg
    li a7, 4
    ecall
    li a0, 3
    li a7, 1
    ecall
    la a0, endl
    li a7, 4
    ecall

    la a0, test_array_3
    la a1, result_array
    mv a2, s0
    jal square_array

    la a0, result_array
    mv a1, s0
    jal print_array

    # Тест 4
    la a0, test_msg
    li a7, 4
    ecall
    li a0, 4
    li a7, 1
    ecall
    la a0, endl
```

```
    li a7, 4
    ecall

    la a0, test_array_4
    la a1, result_array
    mv a2, s0
    jal square_array

    la a0, result_array
    mv a1, s0
    jal print_array

    # Тест 5
    la a0, test_msg
    li a7, 4
    ecall
    li a0, 5
    li a7, 1
    ecall
    la a0, endl
    li a7, 4
    ecall

    la a0, test_array_5
    la a1, result_array
    mv a2, s0
    jal square_array

    la a0, result_array
    mv a1, s0
    jal print_array

    # Тест 6
    la a0, test_msg
    li a7, 4
    ecall
    li a0, 6
    li a7, 1
    ecall
    la a0, endl
    li a7, 4
    ecall
```

```
    la s0, array_size_extra
    lw s0, (s0) # s0 = 9

    la a0, test_array_6
    la a1, result_array_extra
    mv a2, s0
    jal square_array

    la a0, result_array_extra
    mv a1, s0
    jal print_array

    # Выход из программы
    li a7, 10
    ecall

# Подпрограмма обработки массивов
square_array:
    addi sp, sp, -8
    sw s3, 0(sp) # сохраняем на стеке
    sw s4, 4(sp)  # сохраняем на стеке

    mv t0, a0  # указатель на A
    mv t1, a1  # указатель на B
    mv t2, a2  # счетчик элементов
    li t3, 0   # индекс текущего элемента

process_loop:
    lw t4, (t0)  # загрузка элемента из A
    mv t5, t4     # копия для умножения
    li t6, 0      # результат умножения
    li s3, 32     # счетчик битов

multiply:
    andi s4, t5, 1
    beqz s4, shift
    add t6, t6, t4
    bltu t6, t4, overflow
    bltz t6, overflow

shift:
    slli t4, t4, 1
    srli t5, t5, 1
```

```
    addi s3, s3, -1
    bnez s3, multiply

    sw t6, (t1)  # сохранение результата в B
    j next

overflow:
    sw zero, (t1)

next:
    addi t0, t0, 4
    addi t1, t1, 4
    addi t3, t3, 1
    blt t3, t2, process_loop

   # Восстанавливаем используемые значения
    lw s3, 0(sp)
    lw s4, 4(sp)
    addi sp, sp, 8
    ret

# Подпрограмма вывода результата
print_array:
    mv t0, a0 # указатель на B
    mv t1, a1 # размер массива
    # Вывод результатов
    la a0, result_msg
    li a7, 4
    ecall
    li t2, 0
output_loop:
    lw a0, (t0)
    li a7, 1
    ecall
    li a0, ' '
    li a7, 11
    ecall
    addi t0, t0, 4
    addi t2, t2, 1
    blt t2, t1, output_loop

    la a0, endl
    li a7, 4
```

```
    ecall
    ret
```

## Результат выполнения программы

```
Тест 1
Площади квадратов: 4 9 25 49 81
Тест 2
Площади квадратов: 0 121 841 1225 144
Тест 3
Площади квадратов: 1089 196 0 0 0
Тест 4
Площади квадратов: 0 1 16 36 64
Тест 5
Площади квадратов: 12321 49284 110889 197136 986049
Тест 6
Площади квадратов: 1 4 9 16 25 36 49 64 81

-- program is finished running (0) --
```

## Программа на 9 баллов

```
# Макрос вывода строки
.macro print_str(%str)
    la a0, %str
    li a7, 4
    ecall
.end_macro

# Макрос чтения int
.macro read_int(%reg)
    li a7, 5
    ecall
    mv %reg, a0
.end_macro

# Макрос вывода int
.macro print_int(%reg)
    mv a0, %reg
```

```
    li a7, 1
    ecall
.end_macro


# Макрос вывода символа
.macro print_char(%char)
    li a0, %char
    li a7, 11
    ecall
.end_macro

# Ввод массива(буквально тот же код из первой программы, просто с
макросами)
.macro input_array(%array, %size, %prompt)
    mv t0, %array
    li t1, 0
    loop1:
        print_str(%prompt)
        read_int(t2)
        sw t2, (t0)
        addi t0, t0, 4
        addi t1, t1, 1
        blt t1, %size, loop1
.end_macro

# Вывода массива(та же функция вывода, которую я обвесил макросами)
.macro print_array(%array, %size, %msg)
    print_str(%msg)
    mv t0, %array
    li t1, 0
    loop2:
        lw t2, (t0)
        print_int(t2)
        print_char(' ')
        addi t0, t0, 4
        addi t1, t1, 1
        blt t1, %size, loop2
    print_char('\n')
.end_macro

# Макрос генерации тестов
.macro generate_test_array(%array, %size)
```

```
    mv t0, %array
    li t1, 0
    li t2, 1000
    loop3:
        li a7, 41  # Системный вызов для получения случайного числа
        ecall
        bgez a0, mod # Для наглядности работал только с неотрицательными
числами
        neg a0, a0
    mod:
        rem a0, a0, t2 # Беру модуль
        sw a0, (t0)
        addi t0, t0, 4
        addi t1, t1, 1
        blt t1, %size, loop3
.end_macro

.data
prompt_size: .asciz "Введите размер массива: "
prompt_elem: .asciz "Введите элемент: "
result_msg: .asciz "Площади квадратов: "
test_msg: .asciz "Тестовый массив: "
test_msg_res: .asciz "Площади тестового массива: "
array_size: .word 0
.align 2
array_A: .space 64 # Выделяем место для 16 элементов (16 * 4 байта)
array_B: .space 64 # Выделяем место для 16 элементов (16 * 4 байта)

.text
.global main

main:
    # Ввод размера массива
    print_str(prompt_size)
    read_int(s0)
    sw s0, array_size, t0  # Сохраняем размер массива

    # Установка указателей на массивы A и B
    la s1, array_A  # s1 = указатель на массив A
    la s2, array_B  # s2 = указатель на массив B

    # Ввод элементов массива A
    input_array(s1, s0, prompt_elem)
```

```
    # Вызов подпрограммы обработки массивов
    mv a0, s1  # адрес массива A
    mv a1, s2  # адрес массива B
    mv a2, s0  # размер массивов
    jal square_array

    # Вывод результата
    print_array(s2, s0, result_msg)

    # Генерация и вывод тестового массива
    generate_test_array(s1, s0)

    mv a0, s1  # адрес массива A
    mv a1, s2  # адрес массива B
    mv a2, s0  # размер массивов
    jal square_array

    print_array(s1, s0, test_msg)
    print_array(s2, s0, test_msg_res)

    # Выход из программы
    li a7, 10
    ecall

# Подпрограмма обработки массивов
square_array:
    addi sp, sp, -8
    sw s3, 0(sp) # сохраняем на стеке
    sw s4, 4(sp)  # сохраняем на стеке

    mv t0, a0  # указатель на A
    mv t1, a1  # указатель на B
    mv t2, a2  # счетчик элементов
    li t3, 0   # индекс текущего элемента

process_loop:
    lw t4, (t0)  # загрузка элемента из A
    mv t5, t4     # копия для умножения
    li t6, 0      # результат умножения
    li s3, 32     # счетчик битов

multiply:
```

```
    andi s4, t5, 1
    beqz s4, shift
    add t6, t6, t4
    bltu t6, t4, overflow
    bltz t6, overflow

shift:
    slli t4, t4, 1
    srli t5, t5, 1
    addi s3, s3, -1
    bnez s3, multiply

    sw t6, (t1)  # сохранение результата в B
    j next

overflow:
    sw zero, (t1)

next:
    addi t0, t0, 4
    addi t1, t1, 4
    addi t3, t3, 1
    blt t3, t2, process_loop

   # Восстанавливаем используемые значения
    lw s3, 0(sp)
    lw s4, 4(sp)
    addi sp, sp, 8
    ret
```

В данной программе все функции ввода и вывода вынесены в макросы, также подпрограмма вывода массива переехала в макросы. Добавлен макрос на генерацию тестов(числа в тесте от 0 до 999), в этой же программе произведен запуск тестового случая для того, чтобы показать его рабочесть.

## Результат работы программы

```
Введите размер массива: 7
Введите элемент: 3
Введите элемент: 4
Введите элемент: 5
Введите элемент: 8
Введите элемент: 0
Введите элемент: 1
Введите элемент: 11
Площади квадратов: 9 16 25 64 0 1 121
Тестовый массив: 663 377 652 804 686 808 38
Площади тестового массива: 439569 142129 425104 646416 470596 652864 1444

-- program is finished running (0) --
```