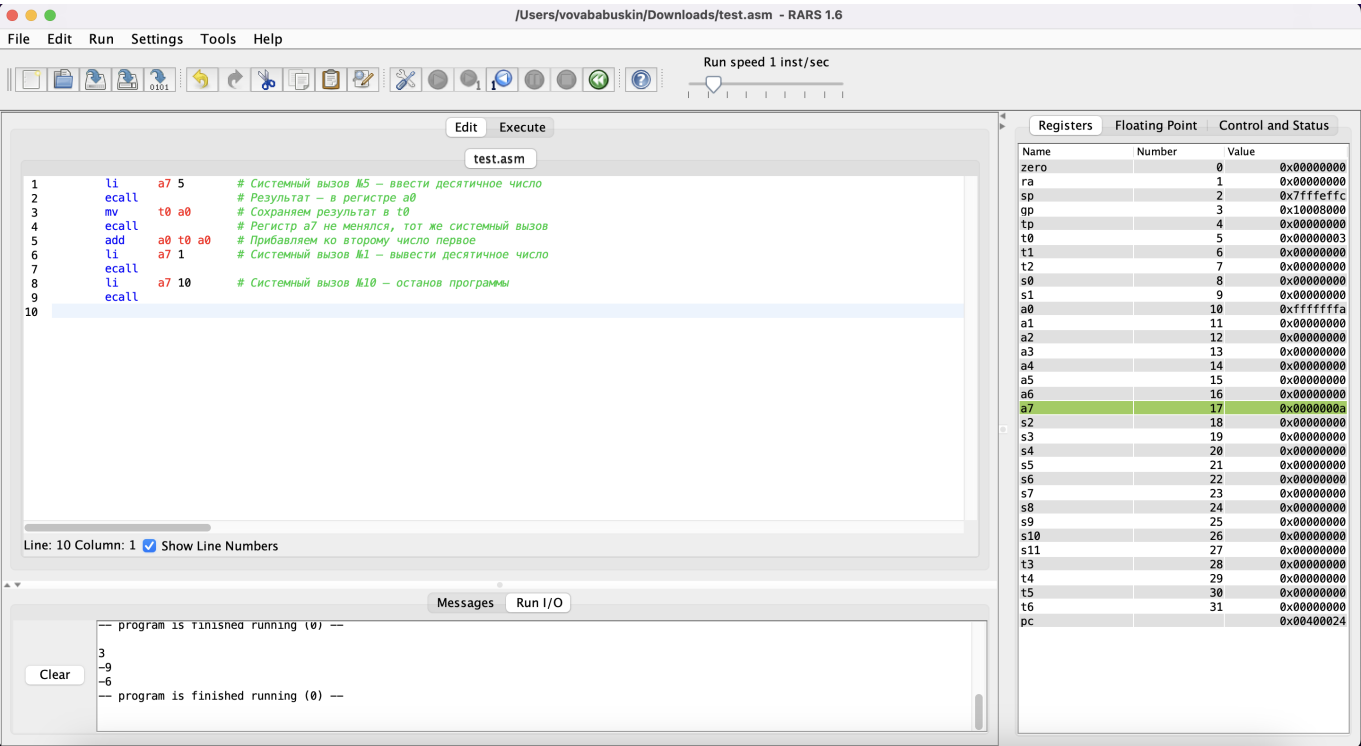


# Бабушкин Владимир Александрович

## АВС ОТЧЕТ 1

### задача 1 A + B

Первым делом я запустил код с семинара, чтобы опробовать систему



Предварительно уменьшил ползунок скорости, чтобы успевать смотреть каждый шаг

Программа выполнялась верно, идем дальше

### задача 2 ВЫВОД

```
1  .text
2      la a0, string      # buffer
3      li a7, 4           # syscall write (4)
4      ecall
5      li a0, 0           # exit code
6      li a7, 10          # syscall exit
7      ecall
8  .data
9      string: .asciz "Hello! It works!!!\n"
10
```

Как я понял:

Секция `.text` обозначает что в этом блоке будет находится сама программа

Секция `.data` обозначает что в этом месте программы будут описываться данные

`.asciz` означает что в конце следующего набора символов нужно добавить `NULL`, тем самым объяснить программе что строка закончена

программа копирует строку в `a0` и выводит ее, после этого очищает `a0` и завершает программу последними двумя строчками.

задача 3 Другой Вывод

```
1  .data
2  hello:
3      .asciz "Hello, world!"
4  .text
5  main:
6      li a7, 4
7      la a0, hello
8      ecall
9
```

здесь мы делаем по сути то же самое, только данные строки поставили в началор программы.

задача 4 Еще один вариант вывода

```
test.asm*
1  .text
2      la a0, string      # buffer
3      li a7, 4           # syscall write (4)
4  .data
5      string: .asciz "Hello! It works!!!\n"
6  .text
7      ecall
8      li a0, 0           # exit code
9      li a7, 10          # syscall exit
10     ecall
11
```

Данные можно поместить и посреди основной программы

задача 5 Вывод на русском

```
test.asm* | 1.asm | 2.asm
1  .text
2      la a0, string      # buffer
3      li a7, 4           # syscall write (4)
4      ecall
5      li a0, 0           # exit code
6      li a7, 10          # syscall exit
7      ecall
8  .data
9      string: .asciz "Привет. Русский язык выглядит так!!!\n"
10
```

RISC-V asm поддерживает и UTF\_16, значит можно выводить строки на русском.

задача 6 Красивое A+B

```

1  .data
2      arg01: .asciz "Input 1st number: "
3      arg02: .asciz "Input 2nd number: "
4      result: .asciz "Result = "
5      ln: .asciz "\n"
6  .text
7      la      a0, arg01      # Подсказка для ввода первого числа
8      li      a7, 4          # Системный вызов №4
9      ecall
10     li      a7 5           # Системный вызов №5 – ввести десятичное число
11     ecall
12     mv      t0 a0          # Сохраняем результат в t0
13
14     la      a0, arg02      # Подсказка для ввода второго числа
15     li      a7, 4          # Системный вызов №4
16     ecall
17     li      a7 5           # Системный вызов №5 – ввести десятичное число
18     ecall
19     mv      t1 a0          # Сохраняем результат в t1
20
21     la a0, result          # Подсказка для выводимого результата
22     li a7, 4              # Системный вызов №4
23     ecall
24     add     a0 t0 t1       # Складываем два числа
25     li      a7 1          # Системный вызов №1 – вывести десятичное число
26     ecall
27
28     la a0, ln              # Перевод строки
29     li a7, 4              # Системный вызов №4
30     ecall
31
32     li      a7 10          # Системный вызов №10 – останов программы
33     ecall
34

```

поделим код на блоки

1–5 данные вывода

7–9 просим пользователя ввести число

10–12 считываем число

14–16 просим ввести второе число

17–19 считываем второе число

21–23 выводим подсказку для результата

24–26 считаем сумму и выводим ее

28–30 переводим строку

32–34 заканчиваем программу