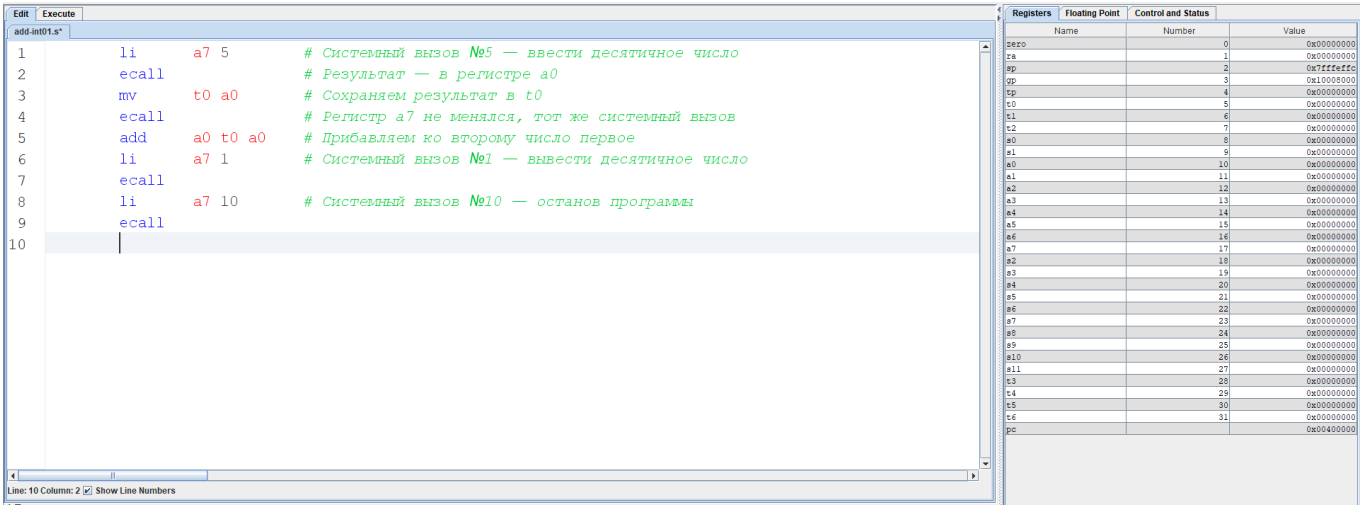
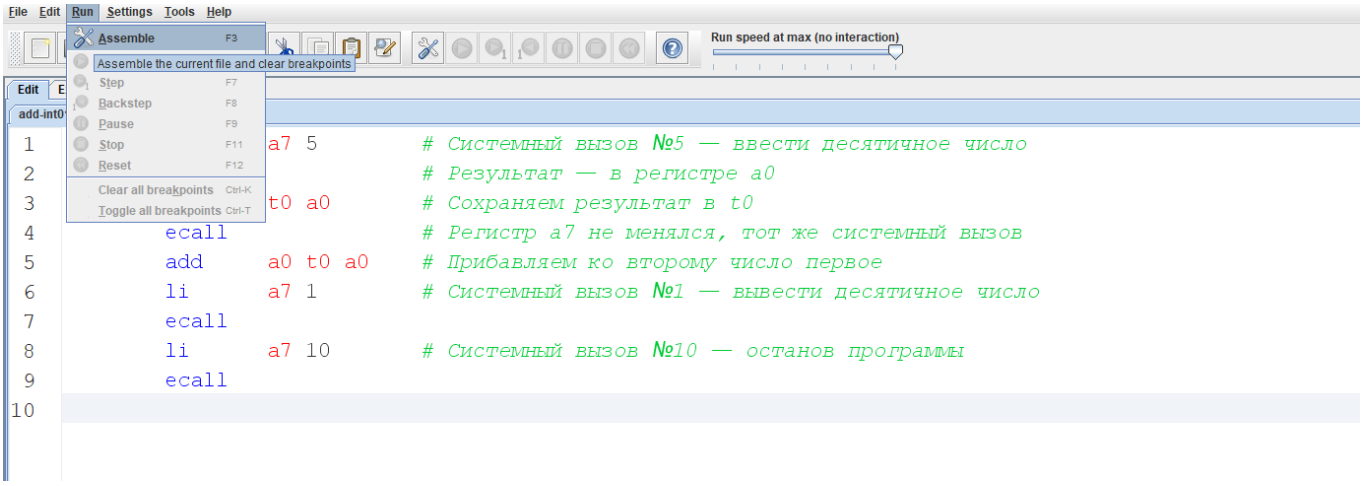
Домашняя работа 1 (Гринченко Евгений, БПИ 236)

Отчёт об использовании эмулятора.

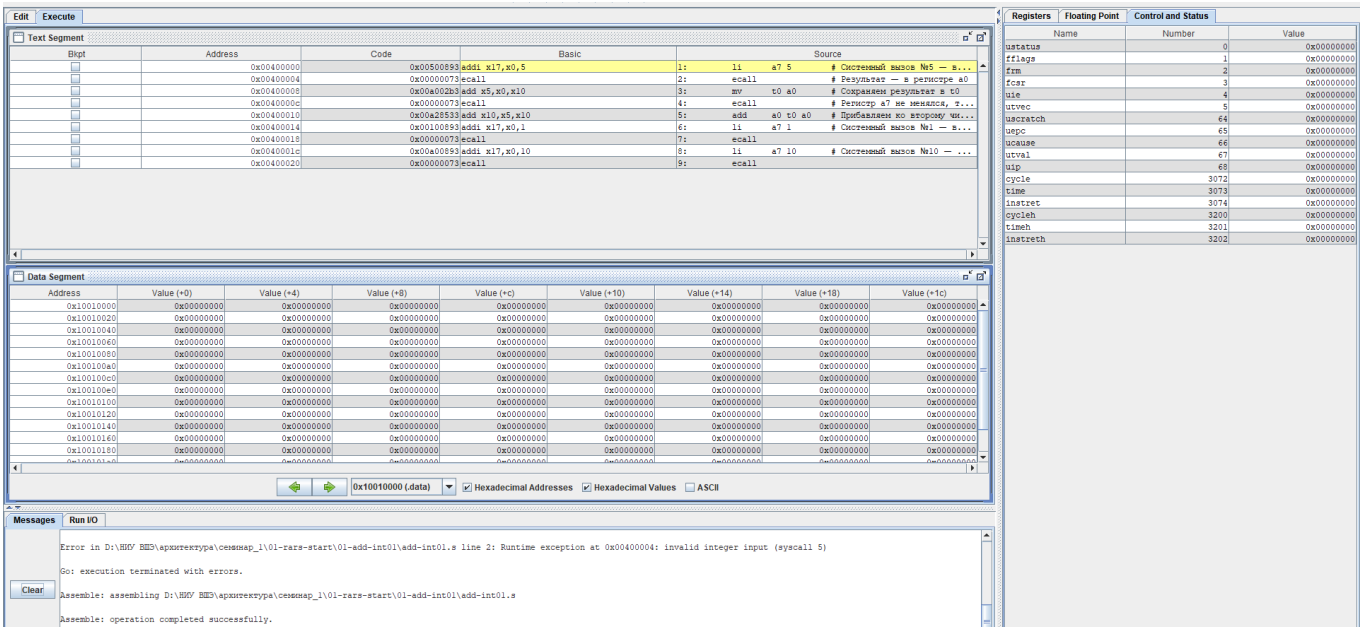
Программа add_int01.s



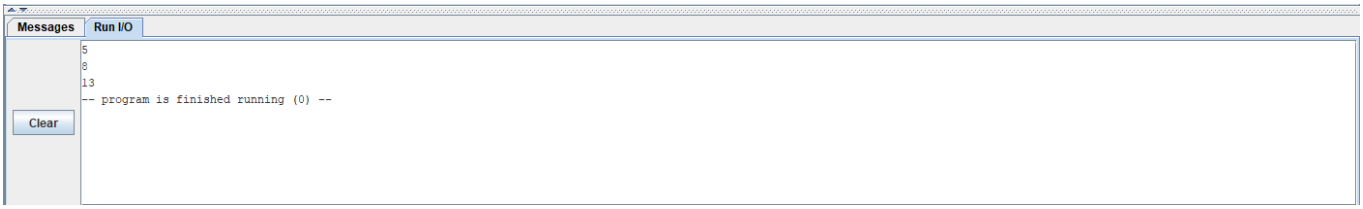
Первым шагом является компиляция файла нашей программы - для этого нужно нажать кнопку Assemble.



После этого мы перейдем во вкладку Execute



Теперь запустим нашу программу и в консоли Run I/O введём произвольные два числа(в нашем случае это будут 5 и 8).

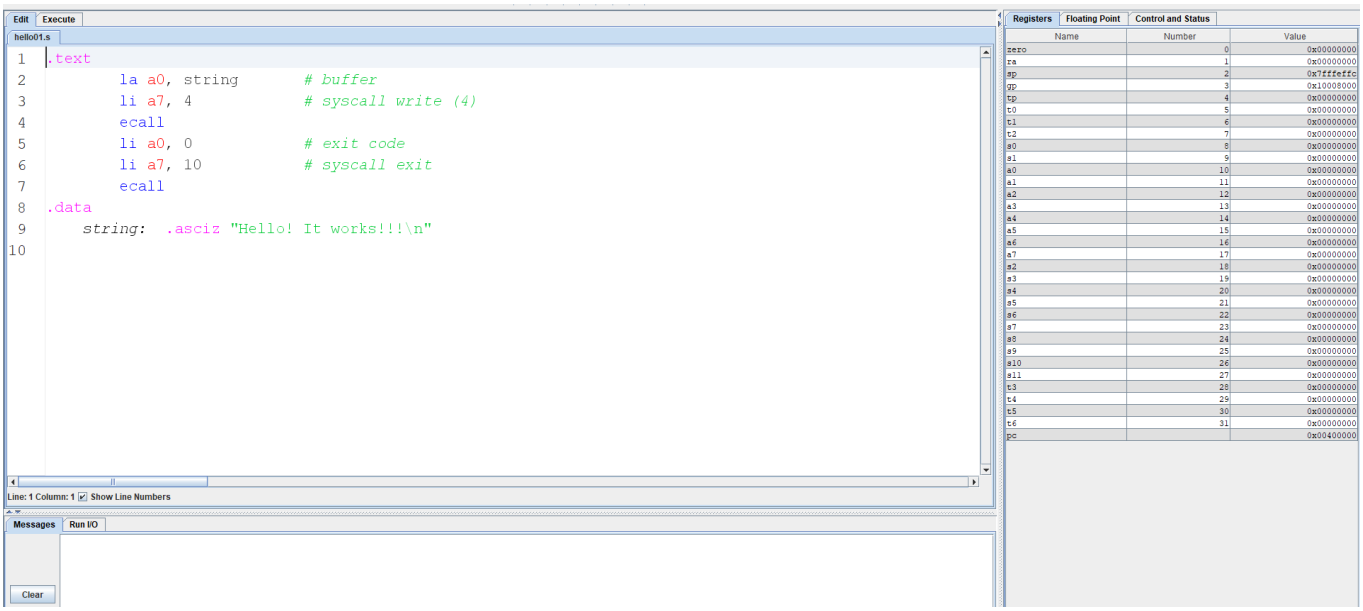


Программа выведет 13 - сумму данных чисел.

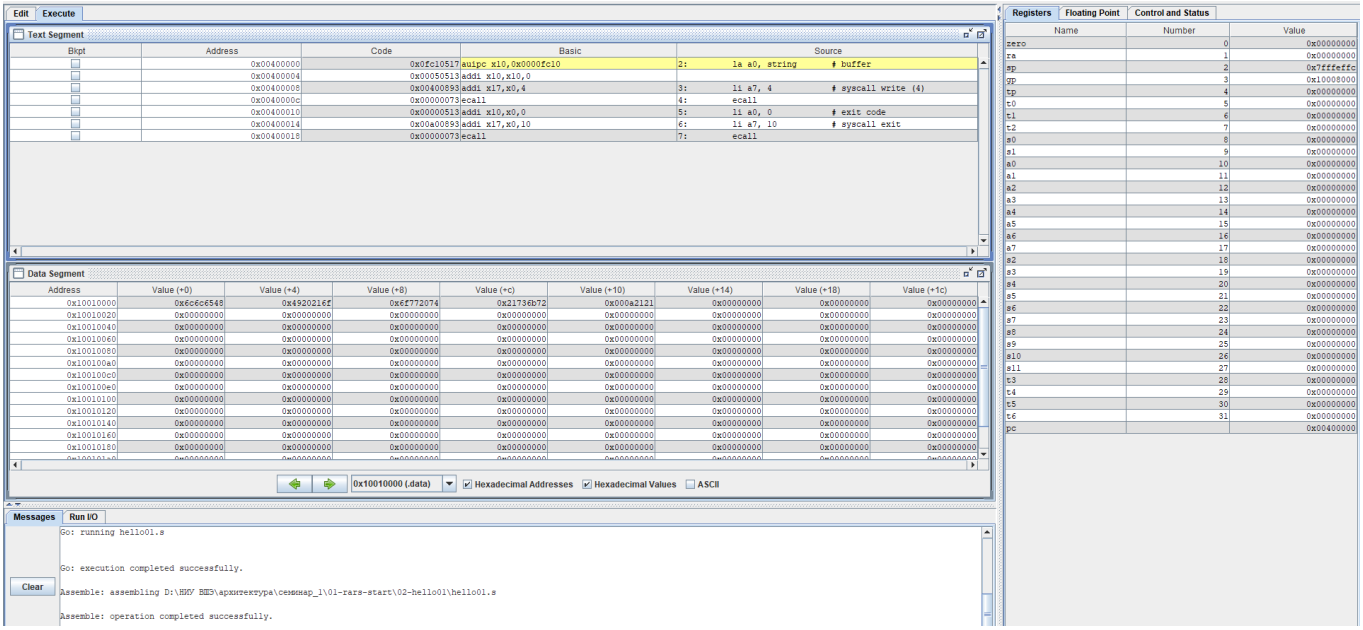
Системные вызовы, которые использовались в add_int01.s

1. Системный вызов №5 - Считывает значение int из консоли ввода и сохраняет результат в регистр a0
2. Системный вызов №1 - Выводит целое число, которое хранится в регистре a0
3. Системный вызов №10 - Завершает работу программы с кодом 0

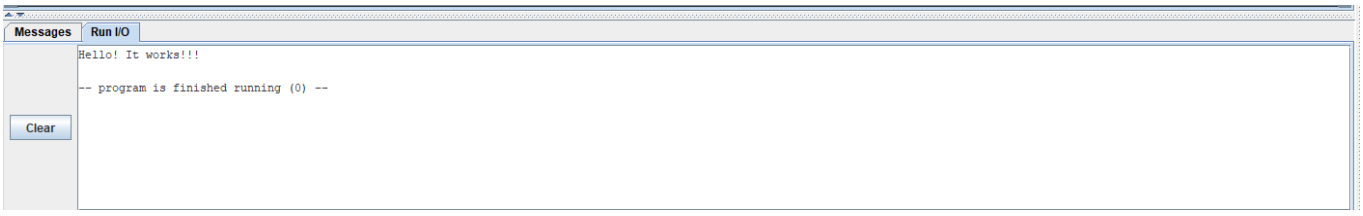
Программа hello01.s



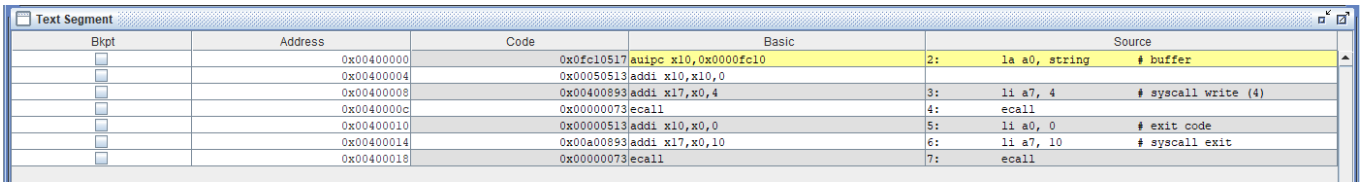
Снова скомпилируем нашу программу и перейдём во вкладку Execute



После этого запустим нашу программу и увидим, что она успешно завершится



Теперь следует провести анализ псевдокоманд, которые есть в нашем коде. Для этого во вкладке Execute обратим внимание на Text Segment



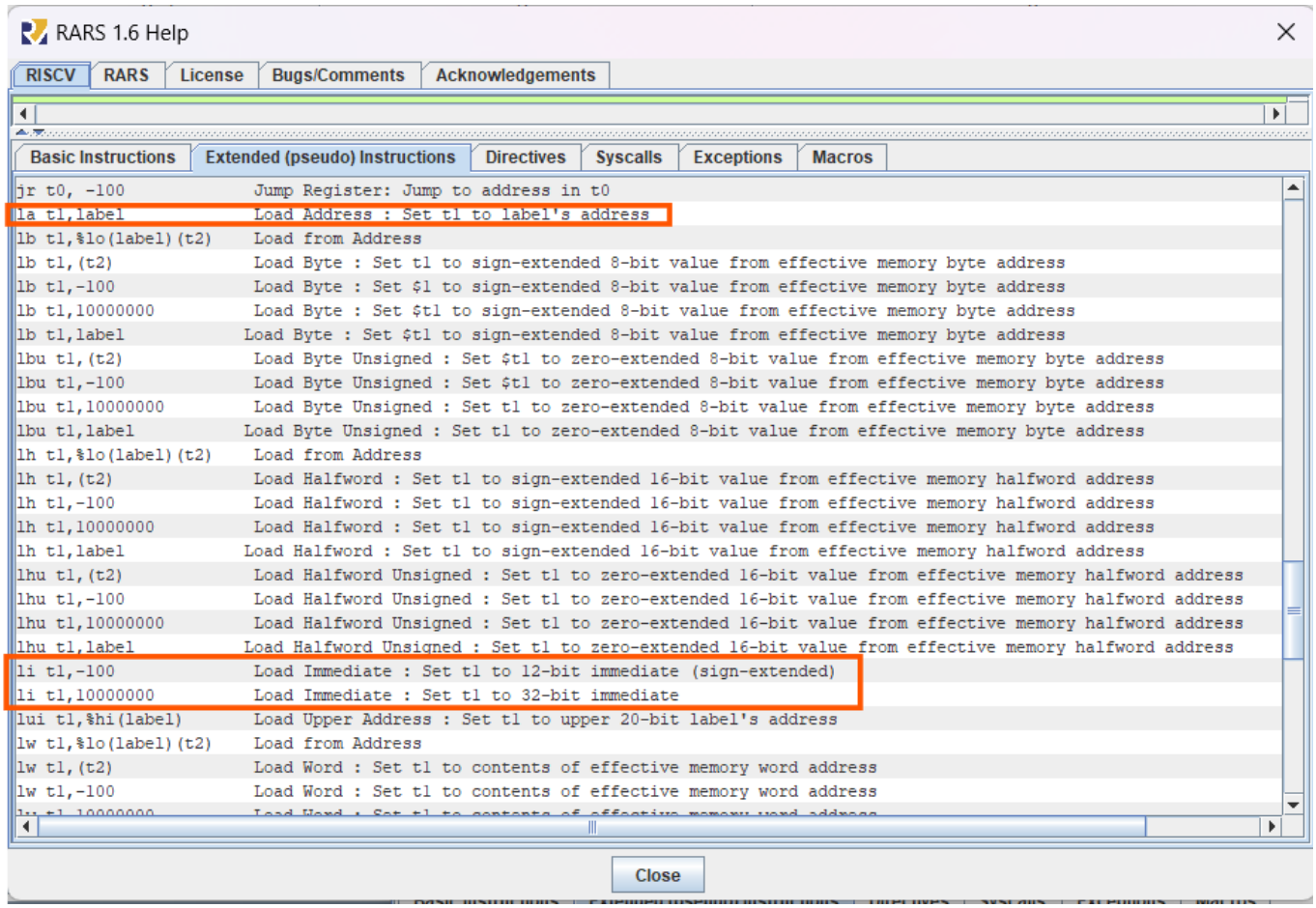
Можем заменить, что у нас в столбце Source находятся строчки нашего исходного кода, а

в столбце Basic для каждой строчки кода указано в какие базовые инструкции происходит переход каждой строчки кода.

Следовательно, можно сделать вывод, что la и li являются псевдоинструкциями, так как, одни и те же строчки кода в Basic и Source не совпадают. В тех случаях, когда строчки в Basic и Source совпадают, мы можем говорить о том, что это базовые инструкции.

Basic	Source		
auipc x10,0x0000fc10	2:	la a0, string	# buffer
addi x10,x10,0			
addi x17,x0,4	3:	li a7, 4	# syscall write (4)
ecall	4:	ecall	
addi x10,x0,0	5:	li a0, 0	# exit code
addi x17,x0,10	6:	li a7, 10	# syscall exit
ecall	7:	ecall	

Действительно, если мы зайдем в справочник эмулятора RARS, то увидим наши команды в разделе Extended Instructions



Директивы:

1. .text - означает, что последующие элементы(инструкции) сохраняются в Text Segment
2. .data - означает, что последующие элементы(инструкции) сохраняются в Data Segment

3. .asciz - означает, что строка сохраняется в Data Segment и после последнего символа обязательно записывается нулевой байт.

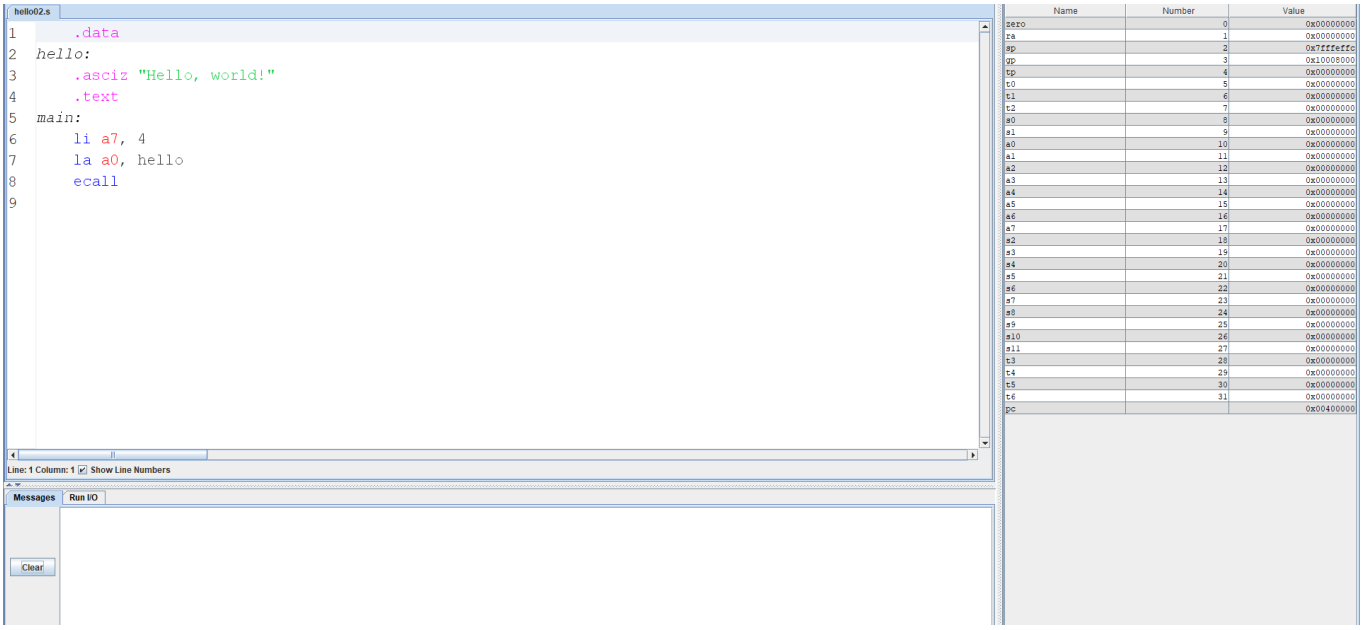
Описание типов форматов команд, используемых в данной программе:

1. li(Load Immediate) - псевдоинструкция, которая преобразуется в базовую инструкцию addi(Addition Immediate, тип «непосредственное значение-регистр-регистр» (Immediate))
2. la(Load Address) - псевдоинструкция, которая преобразуется в базовую инструкцию auipc (Add upper immediate to pc, тип «непосредственное значение-регистр» (Upper))
3. ecall(issue a system call) - базовая инструкция, которая выполняет системный вызов, указанный значением в a7 (тип «непосредственное значение-регистр-регистр» (Immediate)

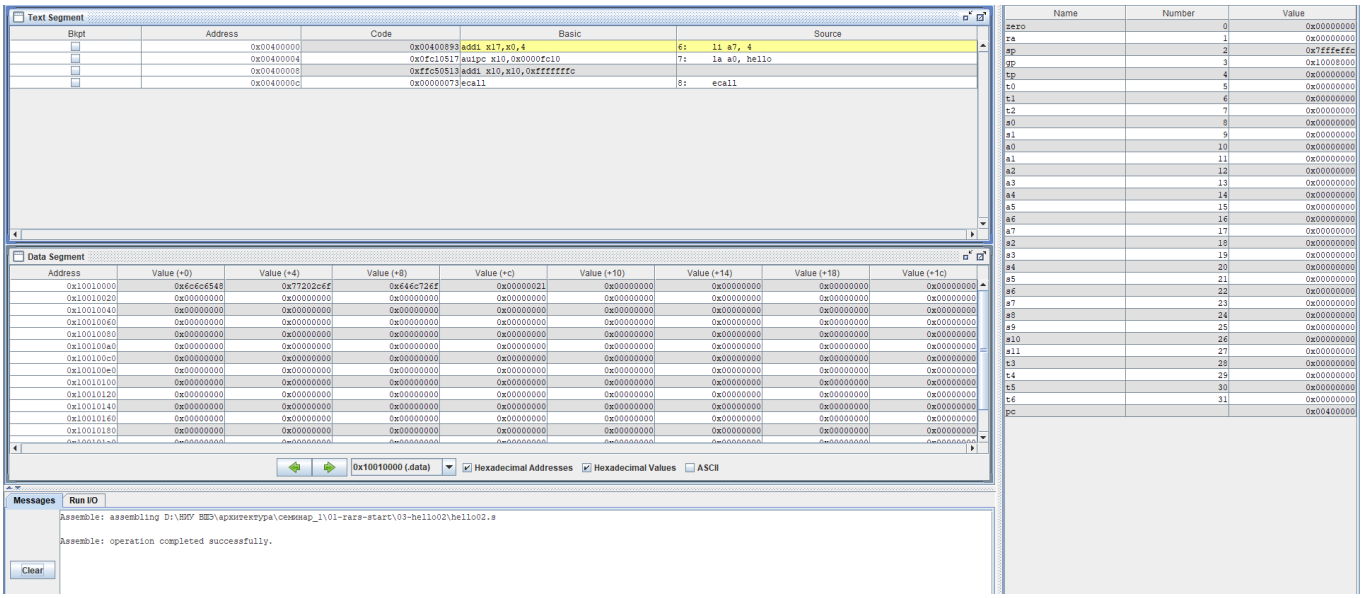
Системные вызовы:

1. Системный вызов №4 - Выводит строку, заканчивающуюся нулем, в консоль (адрес строки хранится в регистре a0)
2. Системный вызов №10 - Завершает работу программы с кодом 0

Программа hello02.s



Скомпилируем нашу программу и перейдём во вкладку Execute



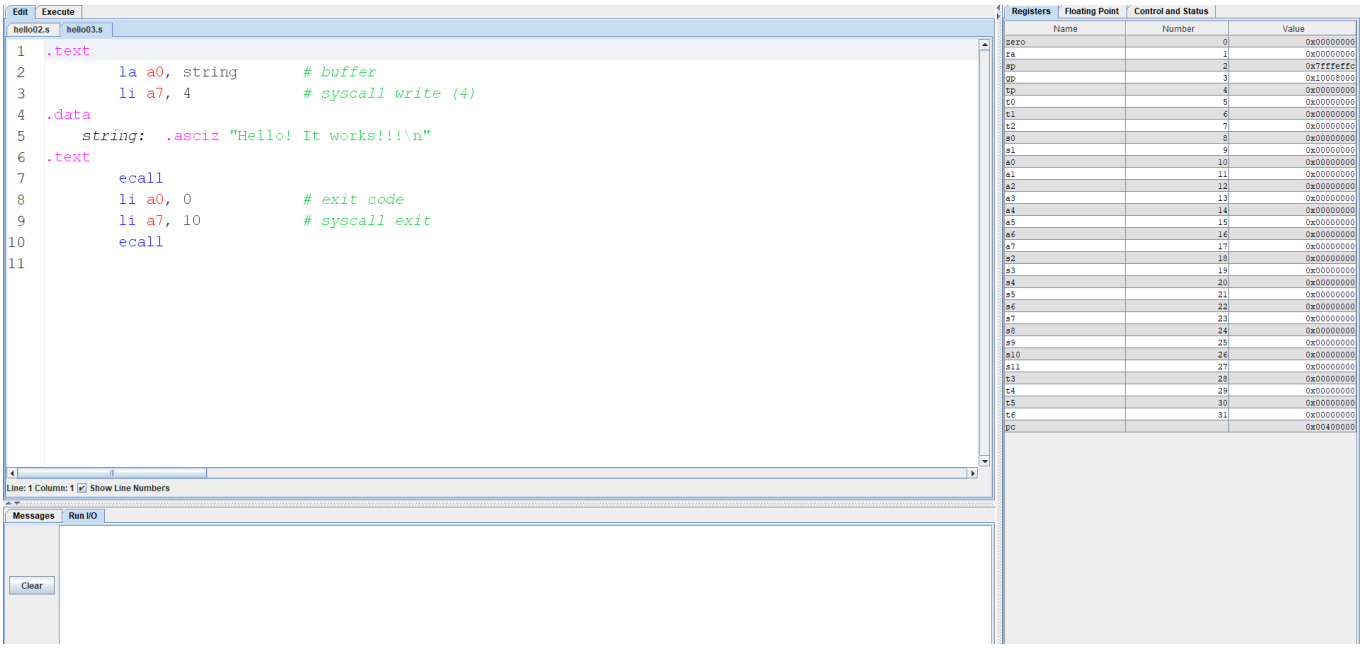
После этого запустим нашу программу

```
Hello, world!
-- program is finished running (dropped off bottom) --
```

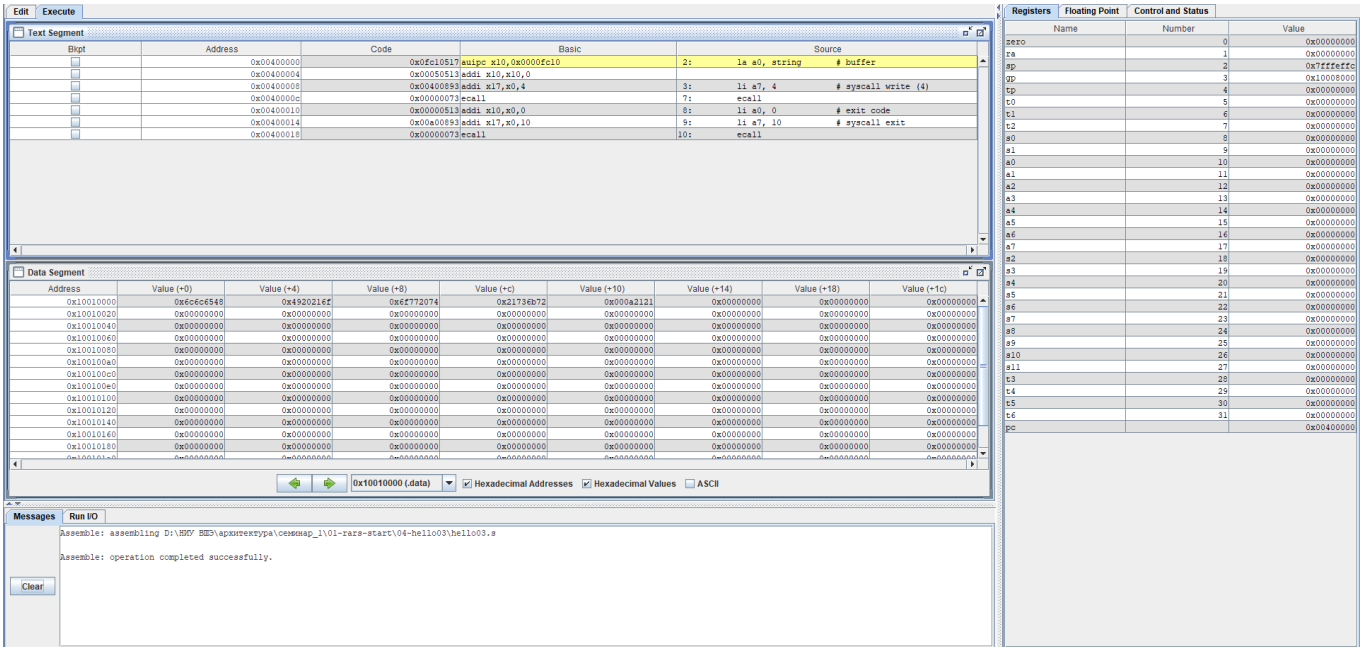
Заметим, что текст вывода отличается от предыдущих двух программ. Именно, программа не завершилась с кодом 0. Это происходит из-за того, что в коде отсутствует системный вызов №10, который отвечает за системный вызов завершения программы. Системные вызовы:

1. Системный вызов №4 - Выводит строку, заканчивающуюся нулем, в консоль (адрес строки хранится в регистре a0)

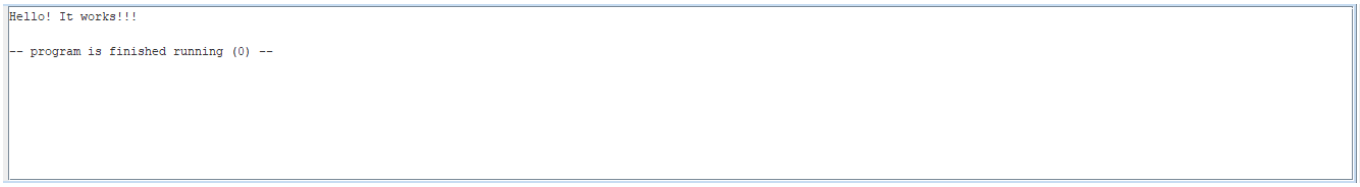
Программа hello03.s



Скомпилируем нашу программу и перейдём во вкладку Execute



Теперь если запустим программу, то увидим такой результат



Также стоит заметить, что несмотря на то, что код секции .text разделен, во вкладке

Execute мы можем увидеть, что он выполняется последовательно.

```
.text
        la a0, string           # buffer
        li a7, 4                # syscall write (4)
.data
    string:  .asciz "Hello! It works!!!\n"
.text
        ecall
        li a0, 0                # exit code
        li a7, 10               # syscall exit
        ecall
```

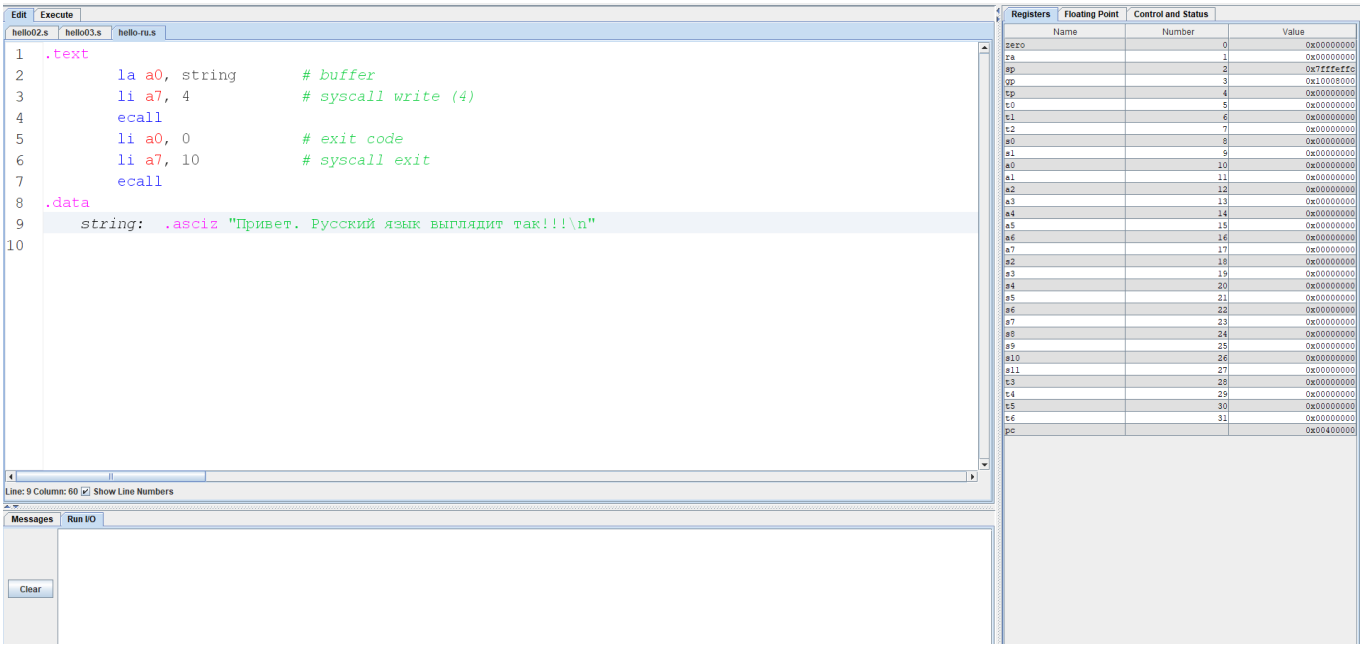
Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x0fc10517	auipc x10,0x0000fc10	2: la a0, string # buffer
	0x00400004	0x00050513	addi x10,x10,0	
	0x00400008	0x00400893	addi x17,x0,4	3: li a7, 4 # syscall write (4)
	0x0040000c	0x00000073	ecall	7: ecall
	0x00400010	0x00000513	addi x10,x0,0	8: li a0, 0 # exit code
	0x00400014	0x00a00893	addi x17,x0,10	9: li a7, 10 # syscall exit
	0x00400018	0x00000073	ecall	10: ecall

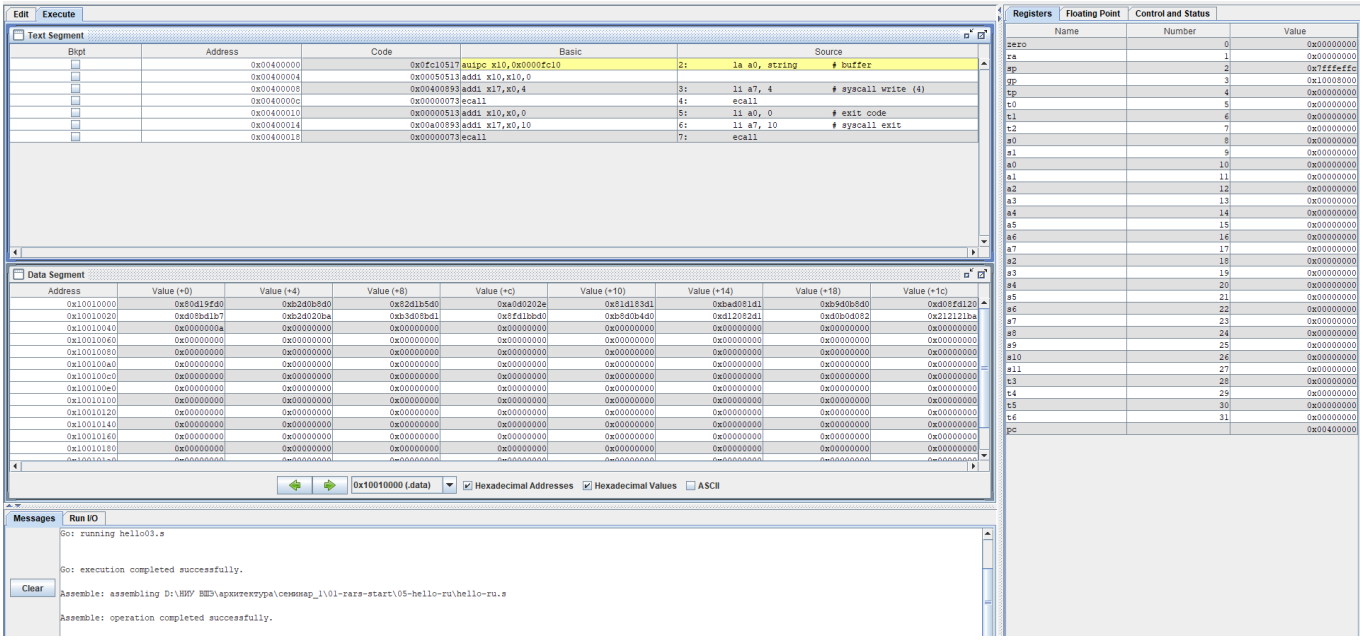
Системные вызовы:

1. Системный вызов №4 - Выводит строку, заканчивающуюся нулем, в консоль (адрес строки хранится в регистре a0)
2. Системный вызов №10 - Завершает работу программы с кодом 0

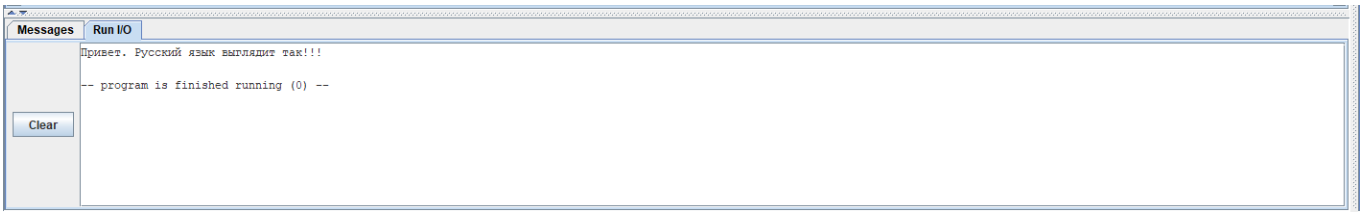
Программа hello-ru.s



Скомпилируем нашу программу и перейдём во вкладку Execute



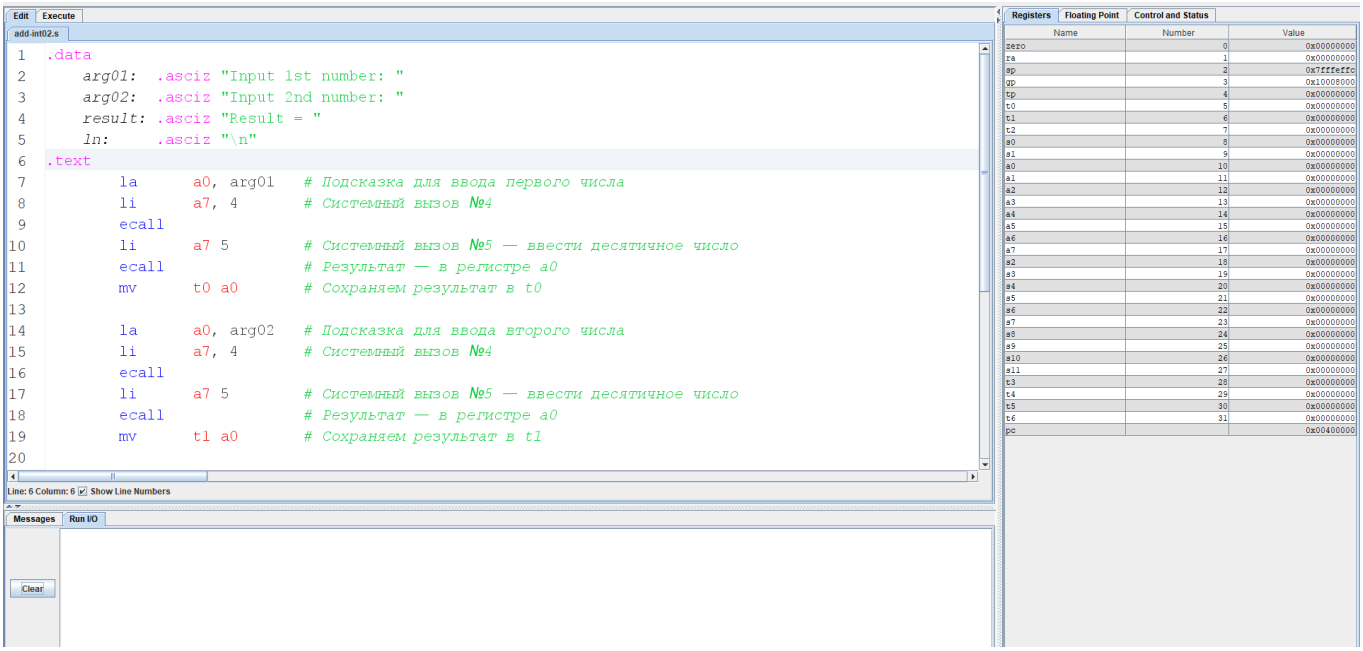
Теперь если запустим программу, то увидим такой результат



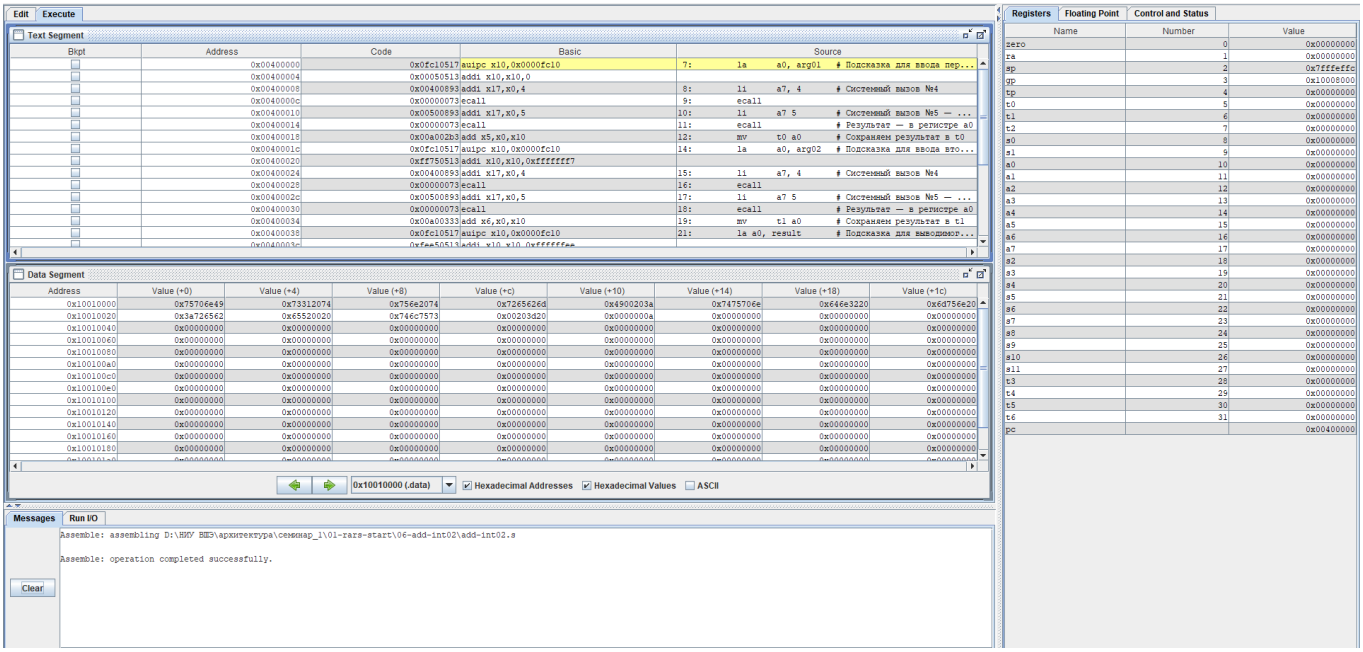
Системные вызовы:

1. Системный вызов №4 - Выводит строку, заканчивающуюся нулем, в консоль (адрес строки хранится в регистре a0)
2. Системный вызов №10 - Завершает работу программы с кодом 0

Программа add-int02.s



Скомпилируем нашу программу и перейдём во вкладку Execute

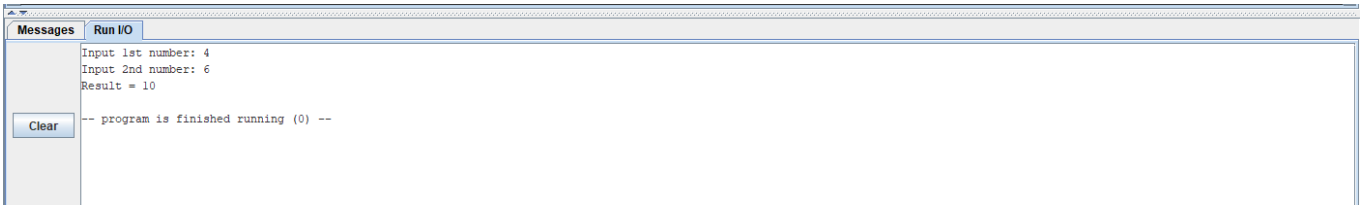


Теперь запустим нашу программу и в консоли Run I/O, увидим, что сначала в консоли появляется строка "Input 1st number: ", потом пользователь должен ввести первое число, далее в консоли появляется строка "Input 2nd number: " и после этого пользователь вводит второе число.

Введём произвольные два числа(в нашем случае это будут 4 и 6).

После этого выведется строка "Result = " и наш итоговый результат сложения (в данном

случае 10)



Системные вызовы:

1. Системный вызов №5 - Считывает значение int из консоли ввода и сохраняет результат в регистр a0
2. Системный вызов №4 - Выводит строку, заканчивающуюся нулем, в консоль (адрес строки хранится в регистре a0)
3. Системный вызов №1 - Выводит целое число, которое хранится в регистре a0
4. Системный вызов №10 - Завершает работу программы с кодом 0