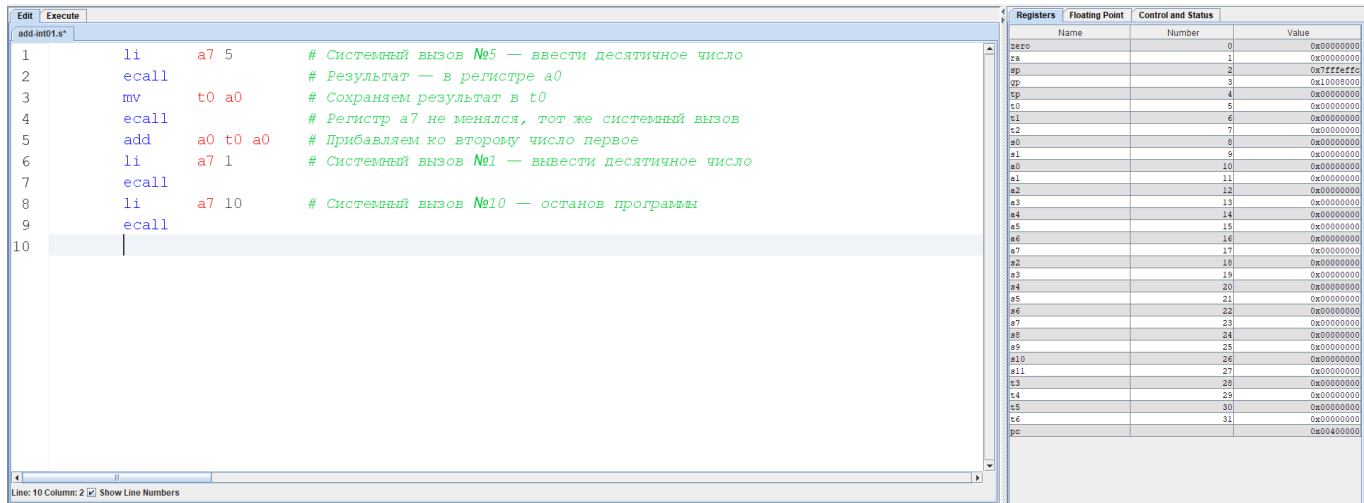


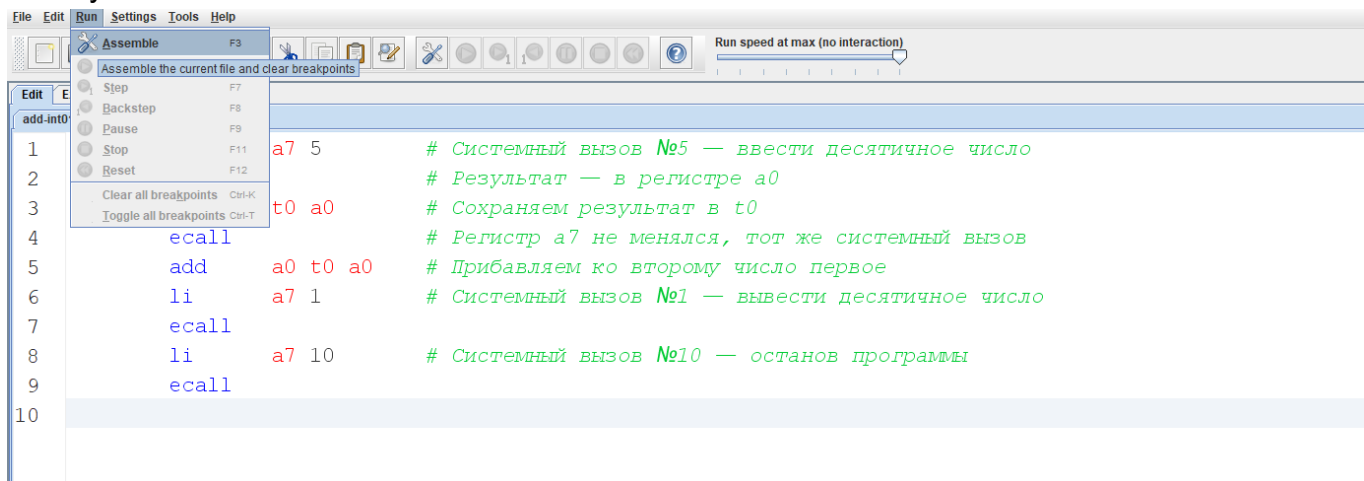
Домашняя работа 1 (Гринченко Евгений, БПИ 236)

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

The screenshot shows the debugger's Execute tab. The Text Segment window displays assembly code with instructions like 'addl \$17,%eax' and 'syscall'. The Data Segment window shows memory addresses and their values. The Registers window on the right lists various registers and their values. The Messages window at the bottom shows an error message: 'Error in D:\BHV\BHV\apnurekrypa\semsnap_1\01-rars-start\01-add-int01.s line 2: Runtime exception at 0x00400004: invalid integer input (syscall 5)'.

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

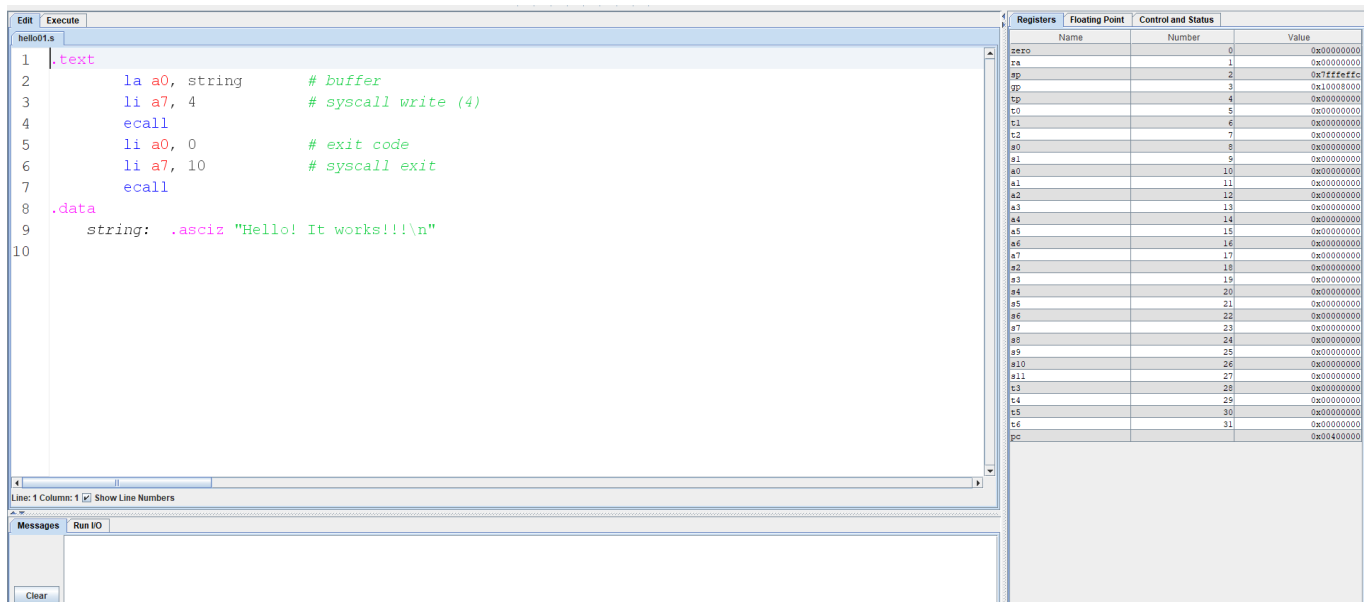
The screenshot shows the debugger's Run I/O window. It shows the input '5' and '8', and the output '13'. The message '-- program is finished running (0) --' is displayed. A 'Clear' button is visible.

Программа выведет 13 - сумму данных чисел.

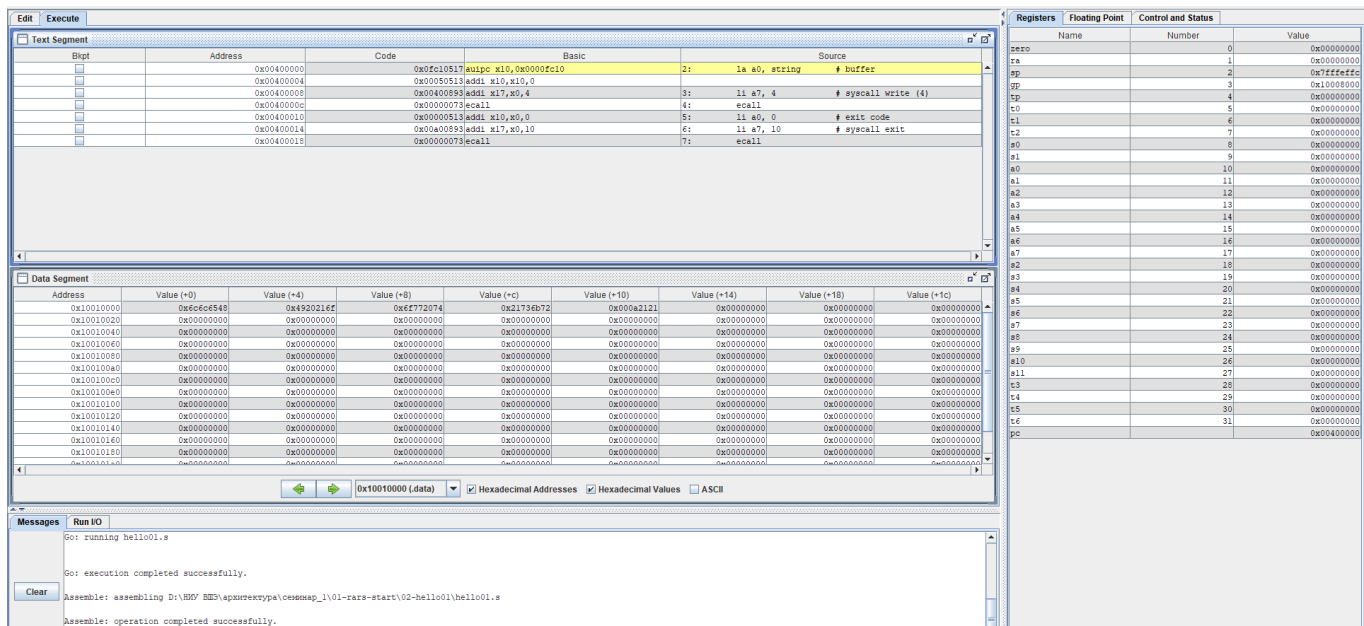
Системные вызовы, которые использовались в add_int01.s

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

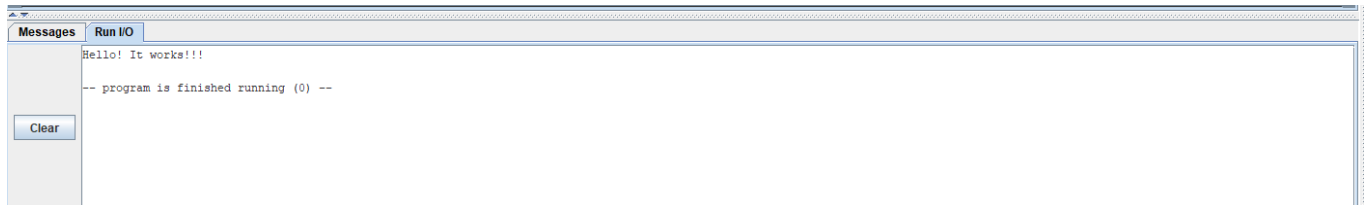
Программа hello01.s



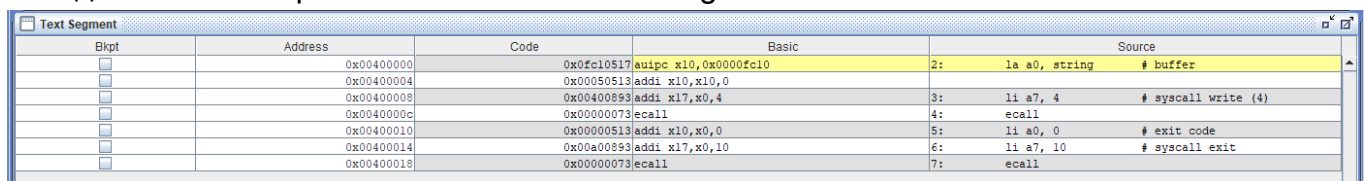
Снова скомпилируем нашу программу и перейдём во вкладку Exeecute



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 Exeecute обратим внимание на Text Segment



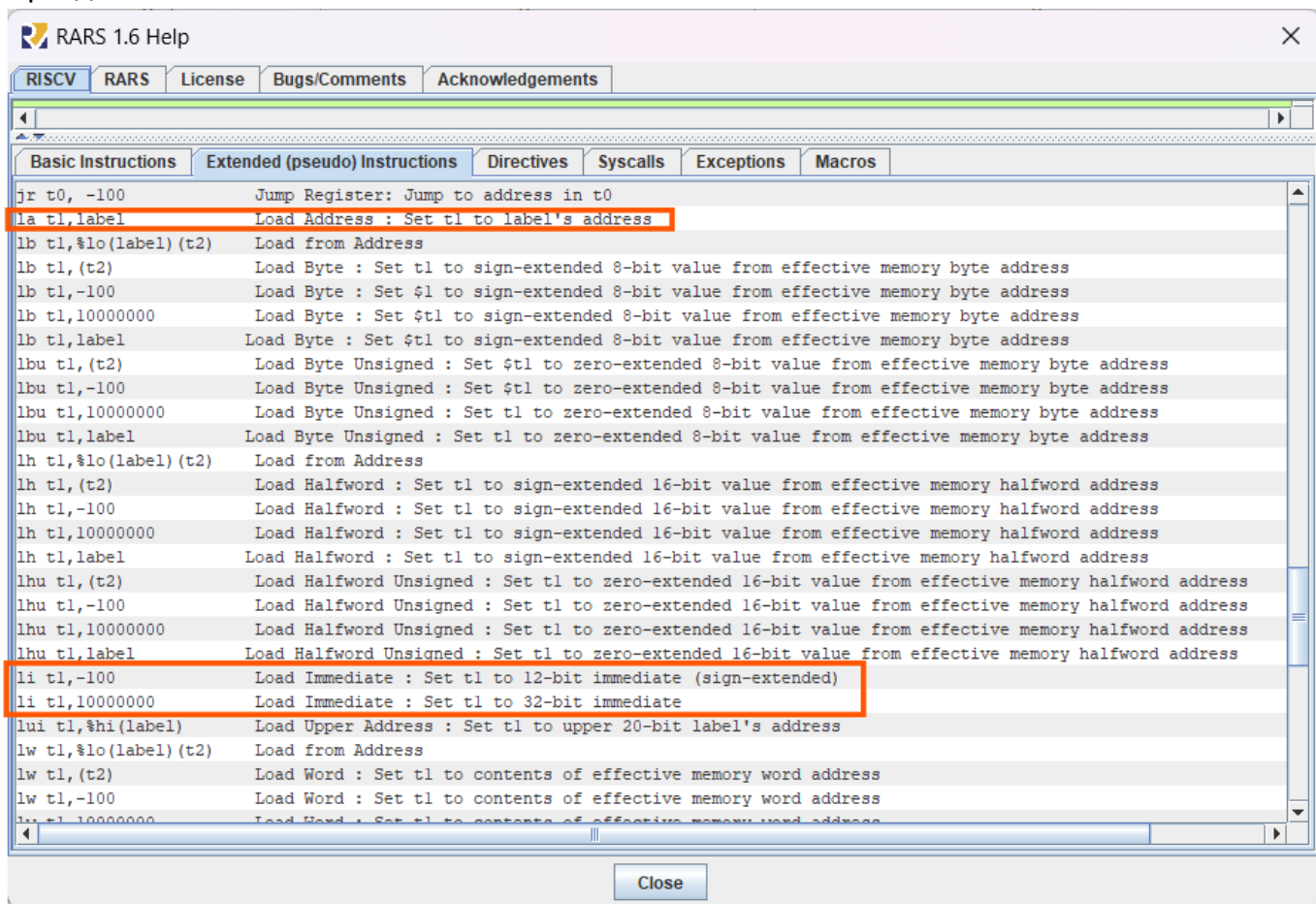
Можем заменить, что у нас в столбце Source находятся строки нашего исходного кода, а

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

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

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

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`

The screenshot shows a MIPS assembler IDE with the following assembly code in the main window:

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

Below the code window, there is a 'Messages' panel with a 'Run I/O' button and a 'Clear' button.

On the right side, there is a register window showing the state of MIPS registers:

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffffff
gp	3	0x00000000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
a0	8	0x00000000
a1	9	0x00000000
a2	10	0x00000000
a3	11	0x00000000
a4	12	0x00000000
a5	13	0x00000000
a6	14	0x00000000
a7	15	0x00000000
s0	16	0x00000000
s1	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
t3	26	0x00000000
t4	27	0x00000000
t5	28	0x00000000
t6	29	0x00000000
pc	31	0x00400000

Скомпилируем нашу программу и перейдём во вкладку Execute

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x00400099	addl %r17,%0_4	6: 11 a7, 4
<input type="checkbox"/>	0x00400004	0x00c10517	bswapl %x10,0x000fc10	7: 1a a0, hello
<input type="checkbox"/>	0x00400008	0xffcc50513	addl %x10,%x10,0xffffffffffc	
<input type="checkbox"/>	0x0040000c	0x000000073	ecall	8: ecall

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

zero

Name

Number

Value

zero		0	0x00000000
ra		1	0x00000000
trp		2	0x7fffffff
%r0		3	0x00000000
%r1		4	0x00000000
%r2		5	0x00000000
%r3		6	0x00000000
%r4		7	0x00000000
%r5		8	0x00000000
%r6		9	0x00000000
%r7		10	0x00000000
%r8		11	0x00000000
%r9		12	0x00000000
%r10		13	0x00000000
%r11		14	0x00000000
%r12		15	0x00000000
%r13		16	0x00000000
%r14		17	0x00000000
%r15		18	0x00000000
%r16		19	0x00000000
%r17		20	0x00000000
%r18		21	0x00000000
%r19		22	0x00000000
%r20		23	0x00000000
%r21		24	0x00000000
%r22		25	0x00000000
%r23		26	0x00000000
%r24		27	0x00000000
%r25		28	0x00000000
%r26		29	0x00000000
%r27		30	0x00000000
%r28		31	0x00000000
%r29			0x00400000

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

2

После этого запустим нашу программу

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:

Программа hello03.s

Ехесите 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"
```

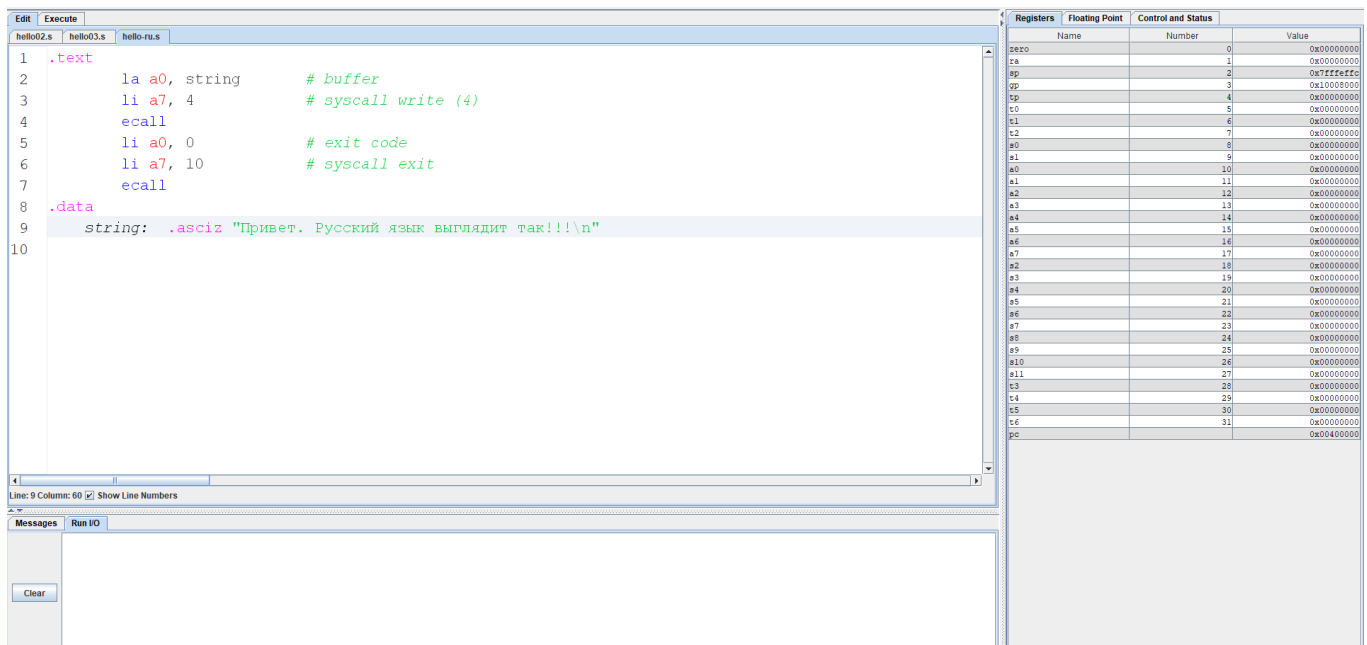
```
.text
    ecall
    li a0, 0           # exit code
    li a7, 10          # syscall exit
    ecall
```

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

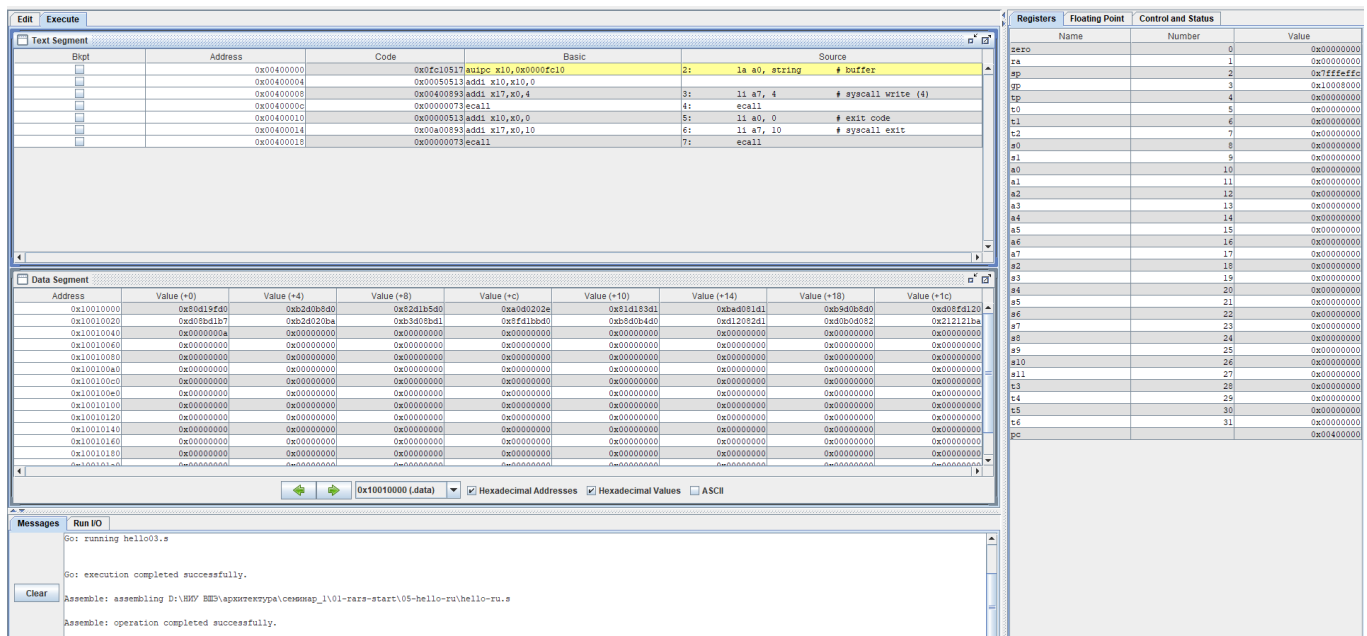
Системные вызовы:

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

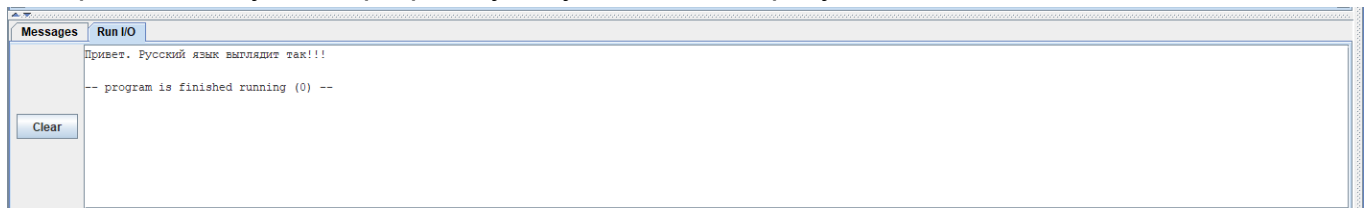
Программа hello-ru.s



Скомпилируем нашу программу и перейдём во вкладку Exeecute



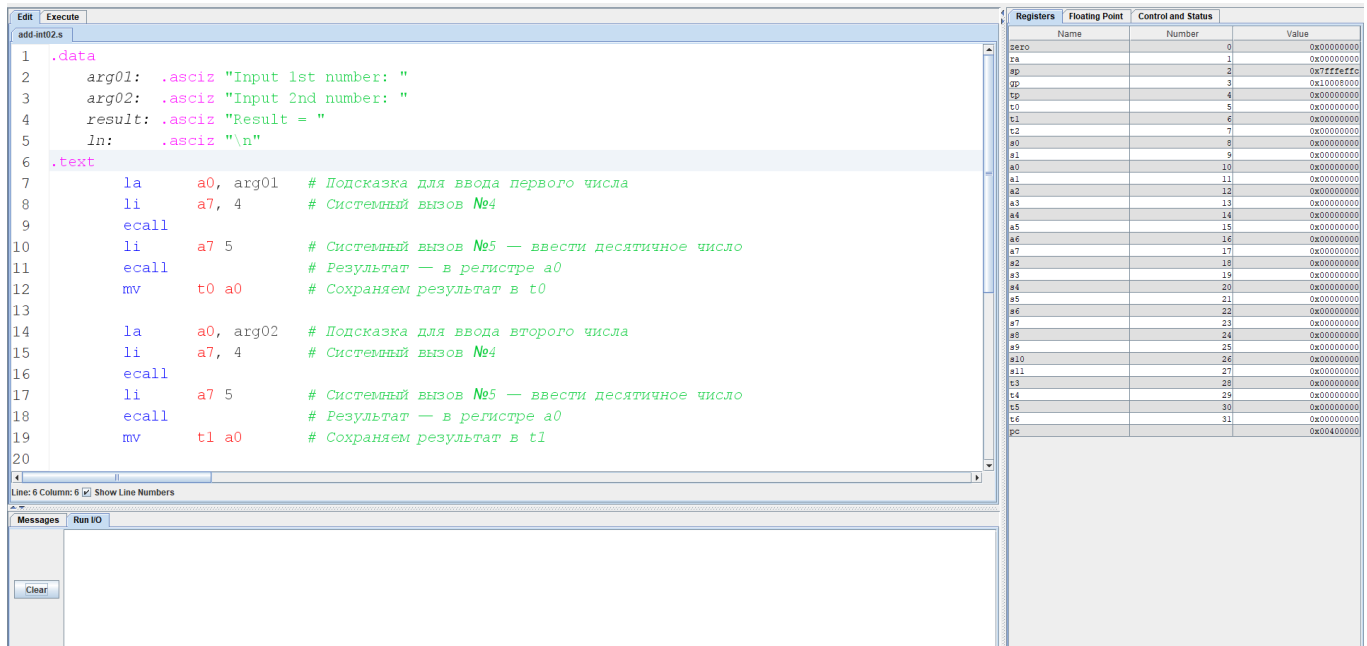
Теперь если запустим программу, то увидим такой результат



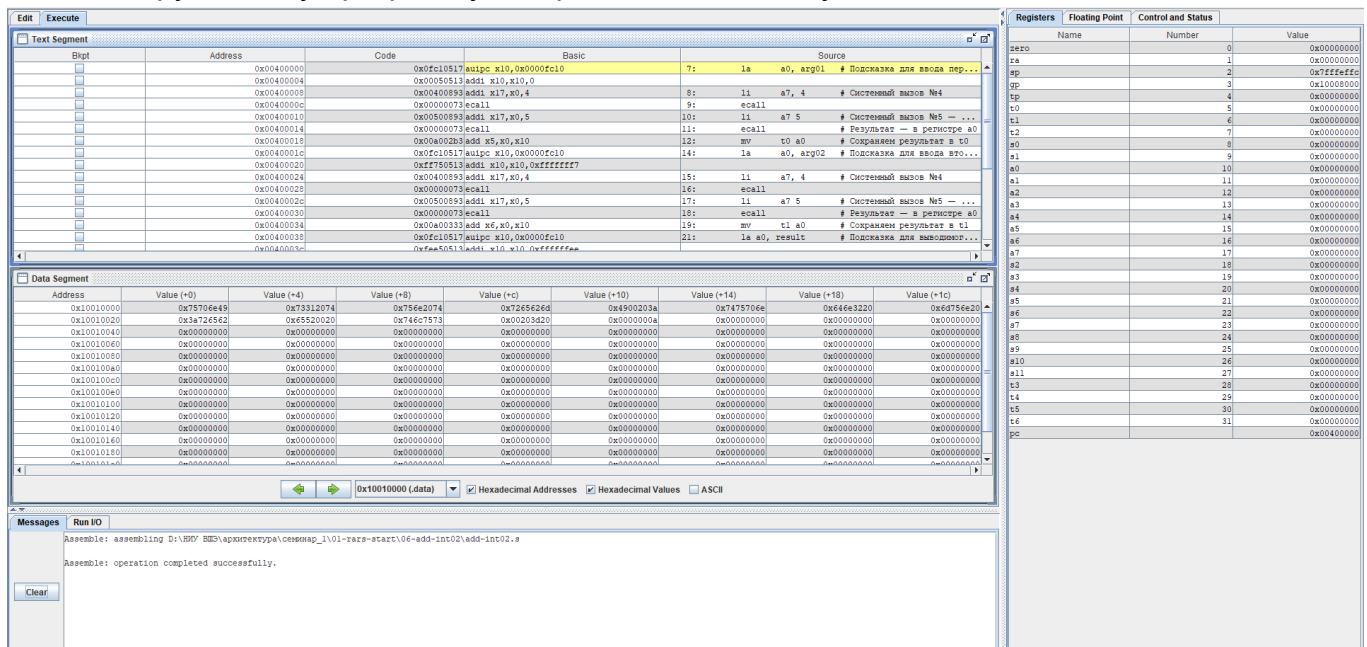
Системные вызовы:

1. Системный вызов №4 - Выводит строку, заканчивающуюся нулем, в консоль (адрес строки хранится в регистре а0)
2. Системный вызов №10 - Завершает работу программы с кодом 0

Программа add-int02.s

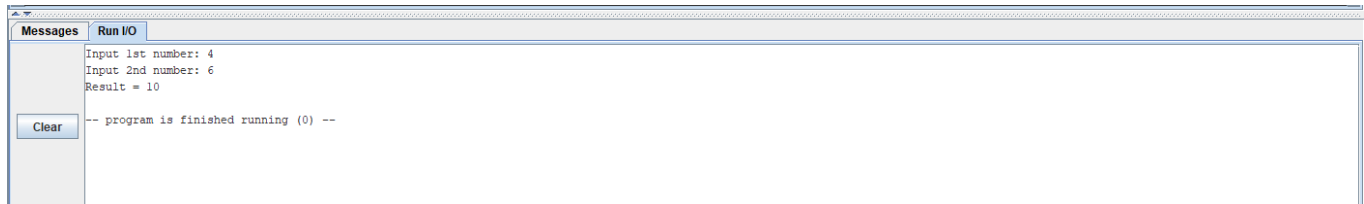


Скомпилируем нашу программу и перейдём во вкладку Exeecute



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