

# Архитектура Вычислительных Систем

Первое домашнее задание

Валерия Чапурина БПИ237

Сентябрь 2024

# Содержание

<b>Первая программа - add-int01.s</b>	<b>3</b>
Запускаем программу . . . . .	3
Псевдокоманды . . . . .	4
Типы форматов команд . . . . .	5
Системные вызовы . . . . .	5
<b>Вторая программа - hello01.s</b>	<b>6</b>
Запускаем программу . . . . .	6
Системные вызовы . . . . .	6
<b>Третья программа - hello02.s</b>	<b>7</b>
Запускаем программу . . . . .	7
Системные вызовы . . . . .	7
<b>Четвёртая программа - hello03.s</b>	<b>8</b>
Запускаем программу . . . . .	8
Системные вызовы . . . . .	8
<b>Пятая программа - hello-ru.s</b>	<b>9</b>
Запускаем программу . . . . .	9
Системные вызовы . . . . .	9
<b>Шестая программа - add-int02.s</b>	<b>10</b>
Запускаем программу . . . . .	10
Системные вызовы . . . . .	10

## Первая программа - add-int01.s

### Запускаем программу

**add-int01.s**

```
1      li      a7 5          # Системный вызов №5 — ввести десятичное число
2      ecall                     # Результат — в регистре a0
3      mv      t0 a0         # Сохраняем результат в t0
4      ecall                     # Регистр a7 не менялся, тот же системный вызов
5      add     a0 t0 a0       # Прибавляем ко второму число первое
6      li      a7 1          # Системный вызов №1 — вывести десятичное число
7      ecall
8      li      a7 10         # Системный вызов №10 — остановка программы
9      ecall
```

Line: 10 Column: 1 ☒ Show Line Numbers

**Messages** **Run I/O**

1000  
2000  
3000

Clear

— program is finished running (0) —

Псевдокоманды

RARS 1.6 Help

RISCv

RARS

License

Bugs/Comments

Acknowledgements

Operand Key for Example Instructions

label, target

any textual label

t1, t2, t3

any integer register

f2, f4, f6

even-numbered floating point register

f8, f6, f8

any floating point register

Basic Instructions

Extended (pseudo) Instructions

Directives

Syscalls

Exceptions

Macros

lhu t1, (t2)

Load Halfword Unsigned : Set t1 to zero-extended 16-bit value from effective memory halfword address

lhu t1, -100

Load Halfword Unsigned : Set t1 to zero-extended 16-bit value from effective memory halfword address

lhu t1, 10000000

Load Halfword Unsigned : Set t1 to zero-extended 16-bit value from effective memory halfword address

lhu t1, label

Load Halfword Unsigned : Set t1 to zero-extended 16-bit value from effective memory halfword address

li t1, -100

Load Immediate : Set t1 to 12-bit immediate (sign-extended)

li t1, 10000000

Load Immediate : Set t1 to 32-bit immediate

lui t1, %hi(label)

Load Upper Address : Set t1 to upper 20-bit label's address

lw t1, %lo(label)(t2)

Load from Address

lw t1, (t2)

Load Word : Set t1 to contents of effective memory word address

lw t1, -100

Load Word : Set t1 to contents of effective memory word address

lw t1, 10000000

Load Word : Set t1 to contents of effective memory word address

lw t1, label

Load Word : Set t1 to contents of memory word at label's address

mv t1, t2

MoVe : Set t1 to contents of t2

neg t1, t2

NEGate : Set t1 to negation of t2

nop

NO OPeration

not t1, t2

Bitwise NOT (bit inversion)

rdmval t1

Read from mval

Close

Здесь мы видим, что **li** и **mv** являются псевдокомандами

Edit

Execute

Text Segment

Bkpt	Address	Code	Basic	
<input type="checkbox"/>	0x00400000	0x00500893	addi x17, x0, 5	1: li
<input type="checkbox"/>	0x00400004	0x00000073	ecall	2: eca
<input type="checkbox"/>	0x00400008	0x00a002b1	add x5, x0, x10	3: mv
<input type="checkbox"/>	0x0040000c	0x00000073	ecall	4: eca
<input type="checkbox"/>	0x00400010	0x00a28533	add x10, x5, x10	5: add
<input type="checkbox"/>	0x00400014	0x00100893	addi x17, x0, 1	6: li
<input type="checkbox"/>	0x00400018	0x00000073	ecall	7: eca
<input type="checkbox"/>	0x0040001c	0x00a00893	addi x17, x0, 10	8: li
<input type="checkbox"/>	0x00400020	0x00000073	ecall	9: eca

А тут становится понятно, какие команды выполняются на самом деле вместо псевдокоманд **li** и **mv**

## Типы форматов команд

- **li**(addi) имеет тип **I**
- **mv** имеет тип **R**
- **ecall** имеет тип **I**
- **add** имеет тип **R**

## Системные вызовы

RARS 1.6 Help

RISCV RARS License Bugs/Comments Acknowledgements

**Operand Key for Example Instructions**

label, target any textual label  
 t1, t2, t3 any integer register  
 f2, f4, f6 even-numbered floating point register

Basic Instructions Extended (pseudo) Instructions Directives **Syscalls** Exceptions Macros

**Table of Available Services**

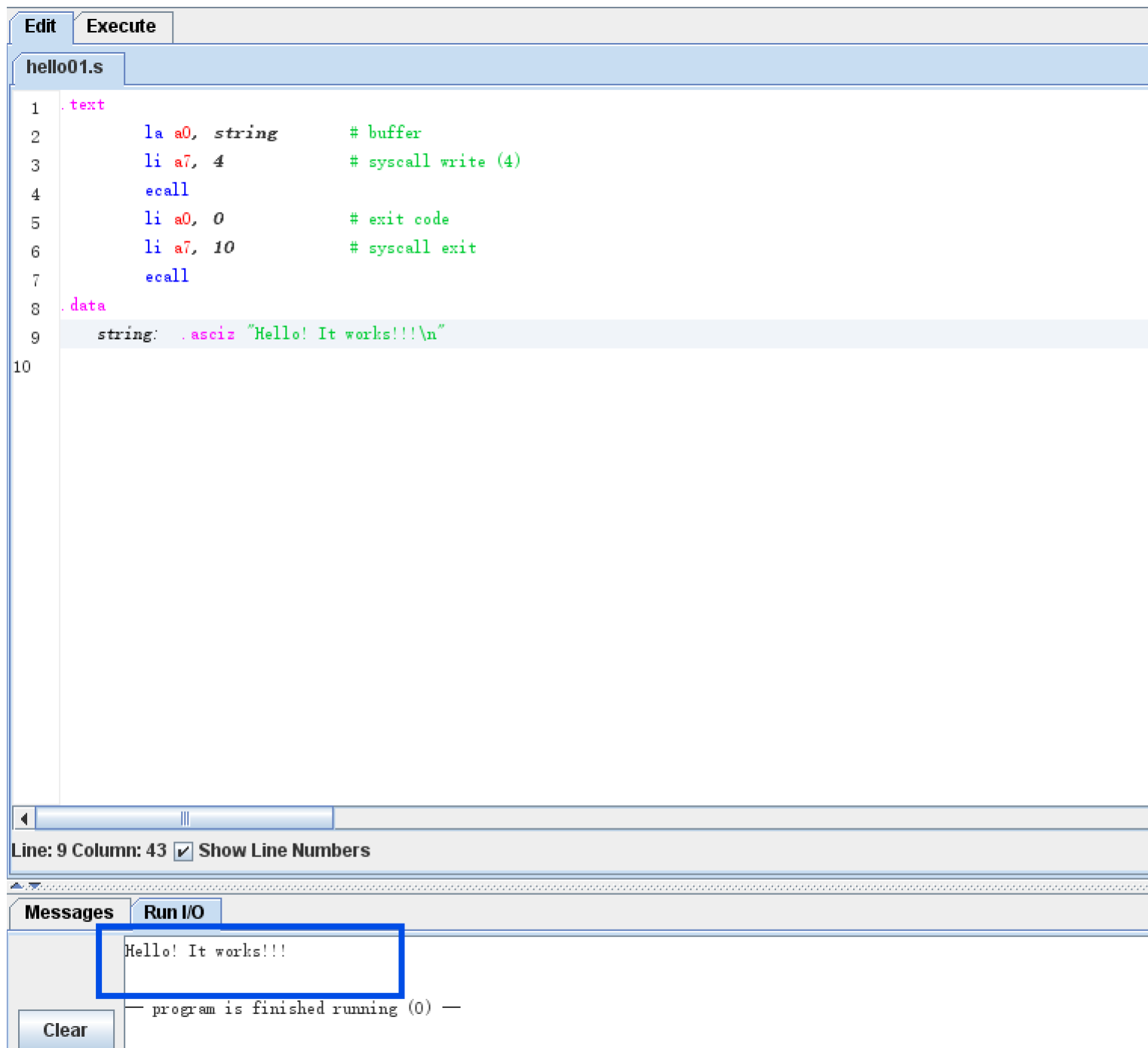
Name	Number	Description	Inputs	Outputs
PrintInt	1	Prints an integer	a0 = integer to print	N/A
PrintFloat	2	Prints a floating point number	fa0 = float to print	N/A
PrintDouble	3	Prints a double precision floating point number	fa0 = double to print	N/A
PrintString	4	Prints a null-terminated string to the console	a0 = the address of the string	N/A
ReadInt	5	Reads an int from input console	N/A	a0 = the int
ReadFloat	6	Reads a float from input console	N/A	fa0 = the float
ReadDouble	7	Reads a double from input console	N/A	fa0 = the double

Close

- **1 - PrintInt.** Выводит число
- **5 - ReadInt.** Читает введенное в консоль число
- **10 - Exit.** Завершает программу с кодом 0

## Вторая программа - hello01.s

### Запускаем программу



```
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
```

Line: 9 Column: 43 ☒ Show Line Numbers

**Messages** **Run I/O**

Hello! It works!!!

— program is finished running (0) —

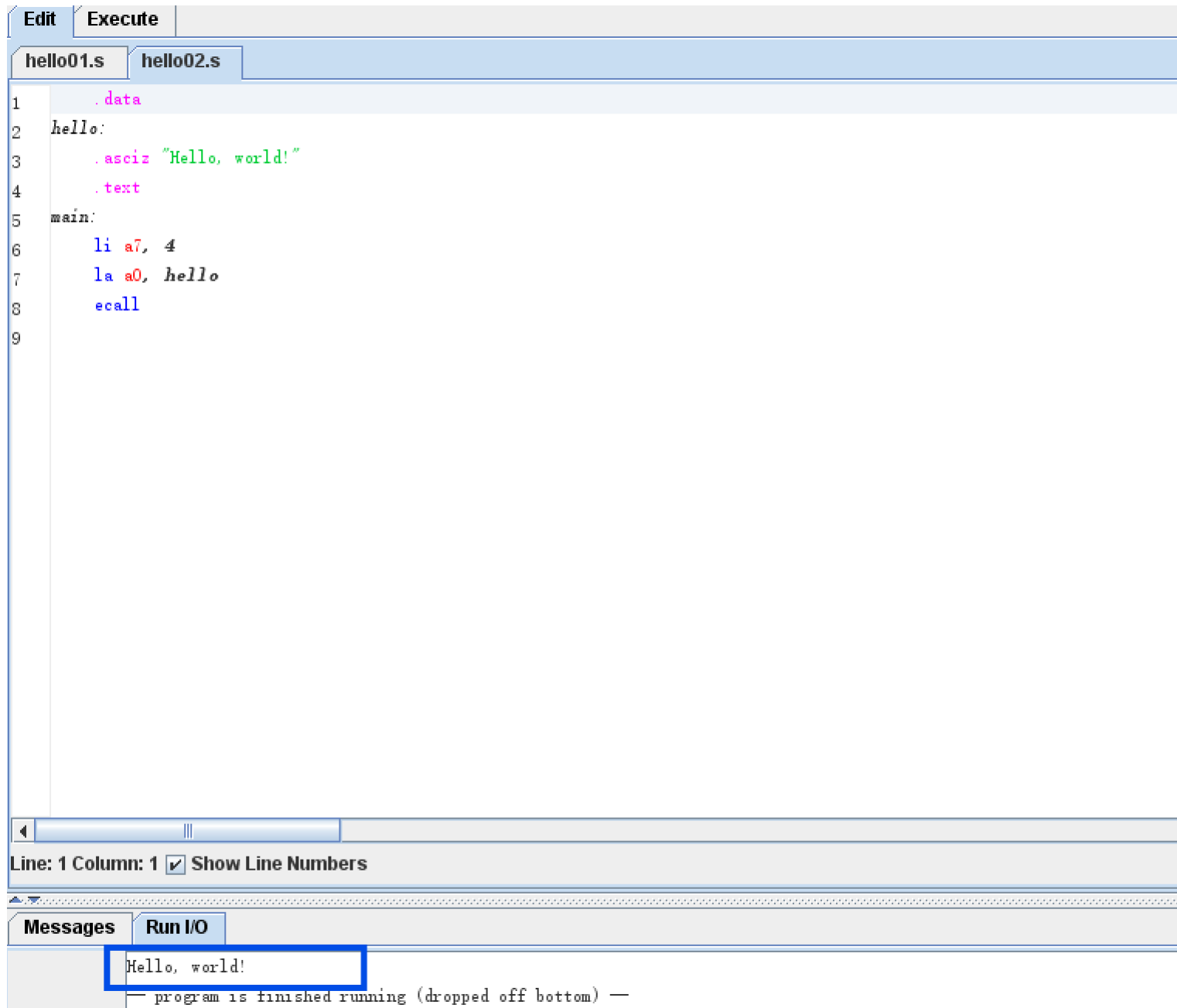
Clear

### Системные вызовы

- 4 - **PrintString**. Выводит в консоль строку, заканчивающуюся нулем
- 10 - **Exit**. Завершает программу с кодом 0

## Третья программа - hello02.s

### Запускаем программу



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

Line: 1 Column: 1 ☒ Show Line Numbers

Messages Run I/O

Hello, world!

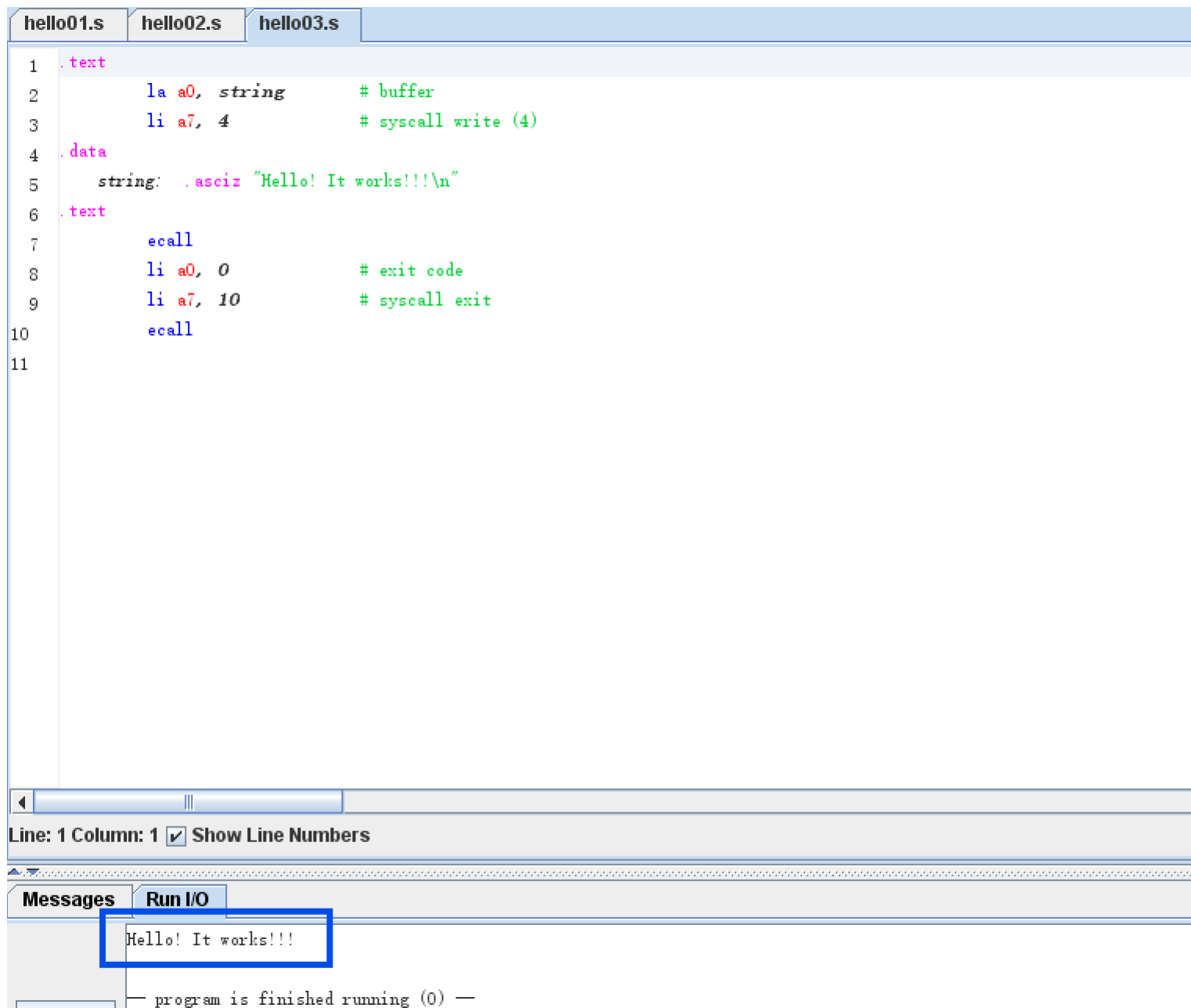
program is finished running (dropped off bottom) —

### Системные вызовы

- 4 - **PrintString**. Выводит в консоль строку, заканчивающуюся нулем

## Четвёртая программа - hello03.s

### Запускаем программу



The screenshot shows a code editor with three tabs: `hello01.s`, `hello02.s`, and `hello03.s`. The `hello03.s` tab is active, displaying the following assembly code:

```
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
```

Below the code editor, the status bar shows "Line: 1 Column: 1" and a checked "Show Line Numbers" option. At the bottom, there are two tabs: "Messages" and "Run I/O". The "Run I/O" tab is active, showing the output "Hello! It works!!!". Below this, a message states "program is finished running (0)".

### Системные вызовы

- 4 - **PrintString**. Выводит в консоль строку, заканчивающуюся нулем
- 10 - **Exit**. Завершает программу с кодом 0



## Пятая программа - hello-ru.s

### Запускаем программу

The screenshot shows a code editor with four tabs: hello01.s, hello02.s, hello03.s, and hello-ru.s. The hello-ru.s tab is active, displaying the following assembly code:

```
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!!!"
10
```

Below the code editor, the status bar indicates "Line: 6 Column: 43" and "Show Line Numbers" is checked. At the bottom, there is a "Messages" panel with a "Run I/O" tab. The "Run I/O" tab shows the output of the program: "Привет. Русский язык выглядит так!!!". Below this, a message states "program is finished running (0)".

### Системные вызовы

- 4 - **PrintString**. Выводит в консоль строку, заканчивающуюся нулем
- 10 - **Exit**. Завершает программу с кодом 0

## Шестая программа - add-int02.s

### Запускаем программу

The screenshot shows a MIPS assembly editor with the file `add-int02.s` selected. The code defines two input strings, `arg01` and `arg02`, and a `result` string. It then uses system calls to prompt for and read two integers, adds them, and prints the result.

```

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               # Результат — в регистре a0
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               # Результат — в регистре a0
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

```

The execution output shows the program's interaction with the user:

```

Input 1st number: 1
Input 2nd number: 2
Result = 3

```

The status bar indicates the program is finished running (0).

### Системные вызовы

- 1 - `PrintInt`. Выводит число
- 4 - `PrintString`. Выводит в консоль строку, заканчивающуюся нулем

- **5 - ReadInt.** Читает введённое в консоль число
- **10 - Exit.** Завершает программу с кодом 0