

# ОТЧЁТ

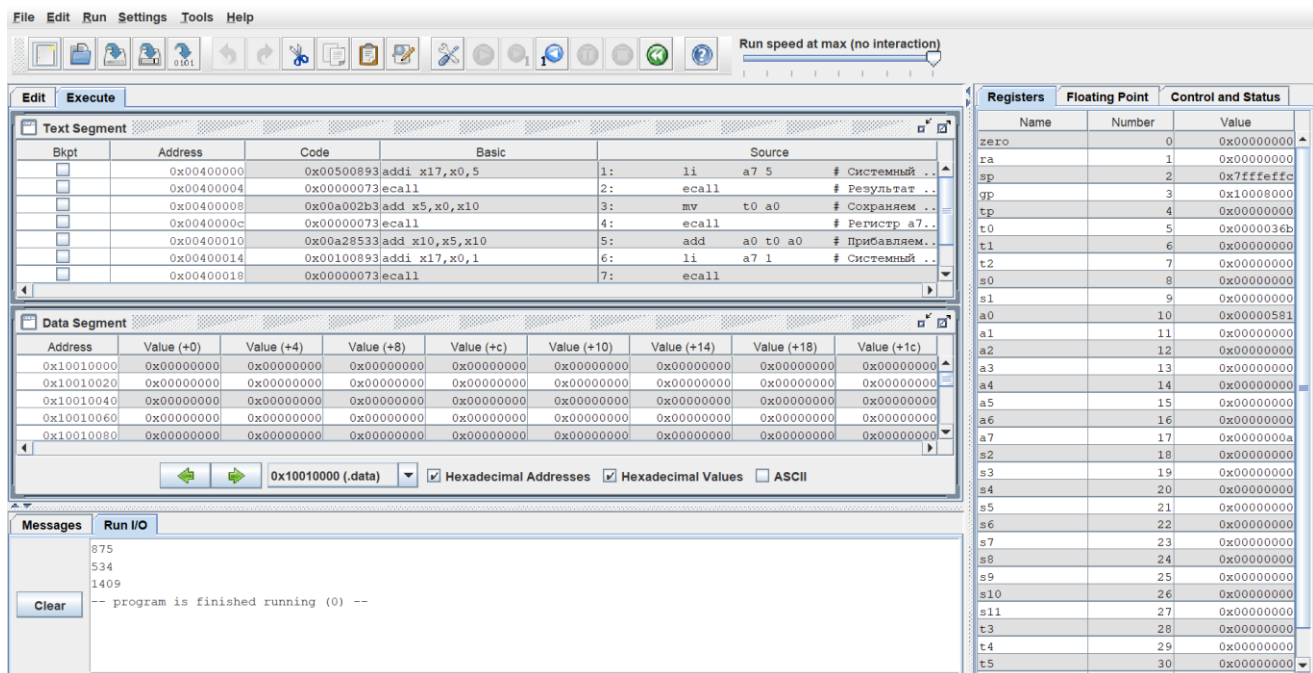
Об использовании эмулятора RARS

Выполнила:  
Студентка 2 курса  
Программной  
инженерии

## 01-add-int01

```
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 — останов программы
9      ecall
```

Результат выполнения программы:



li, mv – псевдокоманды

Типы форматов команд:

- 1) li – псевдокоманда, вызывающая addi(Immediate)
- 2) mv - псевдокоманда, вызывающая add(Register)
- 3) add – R(Register)
- 4) ecall – формат отсутствует

Системные вызовы:

- 1) Системный вызов №5 (ReadInt) – считывание десятичного числа из консоли
- 2) Системный вызов №1 (PrintInt) – вывод десятичного числа в консоль
- 3) Системный вызов №10 (Exit) – завершение программы с кодом 0

## 02-hello01

```
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"
```

Результат выполнения программы:

The screenshot shows the QtConsole application running the '02-hello01' program. The main window displays the assembly code, memory addresses, and the output of the program. The output is 'Hello! It works!!!'.

**Registers**

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffefc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x0000000a
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040001c

Системные вызовы:

- 1) Системный вызов №4 (PrintString) – вывод строки в консоль
- 2) Системный вызов №10 (Exit) – завершение программы с кодом 0

## 03-hello02

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

Результат выполнения программы:

The screenshot displays the QtConsole application interface. The top menu bar includes File, Edit, Run, Settings, Tools, and Help. Below the menu bar is a toolbar with various icons for file operations, editing, and running. A slider for 'Run speed at max (no interaction)' is also present. The main window is divided into three panes: 'Text Segment', 'Data Segment', and 'Registers'. The 'Text Segment' pane shows the assembly code with addresses, codes, and sources. The 'Data Segment' pane shows memory addresses and their values. The 'Registers' pane shows the state of various registers, including 'a0' and 'a7'. The 'Messages' pane at the bottom shows the output 'Hello, world!' and a message indicating the program is finished running.

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffffc0
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x10010000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000004
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400014

Системные вызовы:

- 1) Системный вызов №4 (PrintString) – вывод строки в консоль

## 04-hello03

```
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
```

Результат выполнения программы:

The screenshot displays the MARS MIPS simulator interface. The **Text Segment** window shows the assembly code with addresses and comments. The **Data Segment** window shows memory addresses and values. The **Registers** window lists registers zero through pc with their current values. The **Messages** window shows the output "Hello! It works!!!" and a confirmation message "-- program is finished running (0) --".

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x0000000a
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040001c

Системные вызовы:

- 1) Системный вызов №4 (PrintString) – вывод строки в консоль
- 2) Системный вызов №10 (Exit) – завершение программы с кодом 0

## 05-hello-ru

```
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"
```

Результат выполнения программы:

The screenshot shows the Qt Creator IDE with the following components:

- Text Segment:** A table showing the assembly code for the program. The code includes instructions for loading the string address into register a0, setting register a7 to 4 (syscall write), calling the syscall, setting register a0 to 0 (exit code), setting register a7 to 10 (syscall exit), and calling the syscall again.
- Data Segment:** A table showing the memory layout of the program. It includes the string "Привет. Русский язык выглядит так!!!\n" stored in memory.
- Registers:** A table showing the state of various registers. The registers are listed in two columns: Name, Number, and Value. The registers are zero, ra, sp, gp, tp, t0, t1, t2, s0, s1, a0, a1, a2, a3, a4, a5, a6, a7, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, t3, t4, t5, t6, and pc.
- Messages:** A window showing the output of the program. It displays the message "Привет. Русский язык выглядит так!!!".

Системные вызовы:

- 1) Системный вызов №4 (PrintString) – вывод строки в консоль
- 2) Системный вызов №10 (Exit) – завершение программы с кодом 0

## 06-add-int02

```

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

```

### Результат выполнения программы:

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

☐ Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x0f010517	auipc x10,0x0000f010	7: 1a a0, arg01 # Подсказка...
<input type="checkbox"/>	0x00400004	0x000050513	addi x10,x10,0	
<input type="checkbox"/>	0x00400008	0x00400893	addi x17,x0,4	8: 11 a7, 4 # Системный...
<input type="checkbox"/>	0x0040000c	0x00000073	ecall	9: ecall
<input type="checkbox"/>	0x00400010	0x00500893	addi x17,x0,5	10: 11 a7 5 # Системный...
<input type="checkbox"/>	0x00400014	0x00000073	ecall	11: ecall # Результат...
<input type="checkbox"/>	0x00400018	0x00a002b3	add x5,x0,x10	12: mv t0 a0 # Сохраняем...

☐ Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x75706e45	0x73312074	0x756e2074	0x7265626d	0x4900203a	0x7475706e	0x646e3220	0x6d756e20
0x10010020	0x3a726562	0x65520020	0x746c7573	0x0000000a	0x0000000a	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100f0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Registers Floating Point Control and Status

Name	Number	Value
ustatus	0	0x00000000
fflags	1	0x00000000
frm	2	0x00000000
fcscr	3	0x00000000
uie	4	0x00000000
utvec	5	0x00000000
uscratch	64	0x00000000
uepc	65	0x00000000
ucause	66	0x00000000
utval	67	0x00000000
uip	68	0x00000000
cycle	3072	0x0000001a
time	3073	0x766d457b
instret	3074	0x0000001a
cycleh	3200	0x00000000
timeh	3201	0x0000011a
instreth	3202	0x00000000

Messages Run I/O

Clear

Input 1st number: 15  
Input 2nd number: 37  
Result = 52  
  
-- program is finished running (0) --

## Системные вызовы:

- 1) Системный вызов №4 (PrintString) – вывод строки в консоль
- 2) Системный вызов №5 (ReadInt) – считывание десятичного числа из консоли
- 3) Системный вызов №1 (PrintInt) – вывод десятичного числа в консоль
- 4) Системный вызов №10 (Exit) – завершение программы с кодом 0