

Дз 1

Введение в ассемблер

Творческие изыскания

`a0` – регистр результата

`li a0, 0` – передаем код завершения

`a7` – системный регистр

`li a7, 10` – системное завершение (выход)

`li a7, 1` – системная команда вывода числа (десятичного)

`li a7, 4` – системная команда вывода строки

`li a7, 5` – системная команда для ввода десятичного числа

`.data` – директива блока данных (дата, все логично)

например (этот блок может быть как до, так и после блока `.text`)

```
.data
```

```
name(имя переменной): .asciz "наш текст"
```

`.text` – директива блока кода (текстовый, все логично)

`mv t0, a0` (куда, откуда) `mv` – типа `move`, но на самом деле сложение.

`la a0, (имя переменной)` кладем в регистр результата

ссылка на видео на яндекс диск (+2 балла) https://disk.yandex.ru/i/iIRX7_fcIOAB8A

riscv1.asm*
1add-int01.s
hello01.s
hello02.s*
hello03.s*
hello-ru.s
6add-int02.s

1 .|

.data	Subsequent items stored in Data segment at next available address
.text	Subsequent items (instructions) stored in Text segment at next available address
.word	Store the listed value(s) as 32 bit words on word boundary
.dword	Store the listed value(s) as 64 bit double-word on word boundary
.ascii	Store the string in the Data segment but do not add null terminator
.asciz	Store the string in the Data segment and add null terminator
.string	Alias for .asciz
.byte	Store the listed value(s) as 8 bit bytes
.align	Align next data item on specified byte boundary (0=byte, 1=half, 2=word, 3=double)
.half	Store the listed value(s) as 16 bit halfwords on halfword boundary
.space	Reserve the next specified number of bytes in Data segment
.double	Store the listed value(s) as double precision floating point
.float	Store the listed value(s) as single precision floating point
.extern	Declare the listed label and byte length to be a global data field
.globl	Declare the listed label(s) as global to enable referencing from other files
.global	Declare the listed label(s) as global to enable referencing from other files
.eqv	Substitute second operand for first. First operand is symbol, second operand is expression (like #define)
.macro	Begin macro definition. See .end_macro
.end_macro	End macro definition. See .macro
.include	Insert the contents of the specified file. Put filename in quotes.
.section	Allows specifying sections without .text or .data directives. Included for gcc comparability

Messages

Run I/O

```

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

```

например `.word` псевдокоманда

Напишем свой первый код

```

.data
first_word .asciz

```

```

.asciz  Store the string in the Data segment and add null terminator

```

```

.data
first_word: .asciz "мое первое слово\n"
.text
la a0, first_word
li

```

```

li  Load Immediate

```

```

.data
first_word: .asciz "мое первое слово\n"
.text
la a0, first_word
li a7, 4
ecall

```

```

ecall  Issue a system call : Execute the system call specified by value in a7

```

```
1  .data
2      first_word: .asciz "мое первое слово\n"
3  .text
4      la a0, first_word
5      li a7, 4
6      ecall
7      li a0, 0
8      li a7, 10
9      ecall
```

Line: 9 Column: 7 ☒ Show Line Numbers

мое первое слово

-- program is finished running (0) --

опа, все работает)

но если изменить код в 4 строке на `li a7, 1` (вывод десятичных чисел)

```
1  .data
2      first_word: .asciz "мое первое слово\n"
3  .text
4      la a0, first_word
5      li a7, 1
6      ecall
7      li a0, 0
8      li a7, 10
9      ecall
```

Line: 9 Column: 8 ☒ Show Line Numbers

мое первое слово

-- program is finished running (0) --

268500992

-- program is finished running (0) --

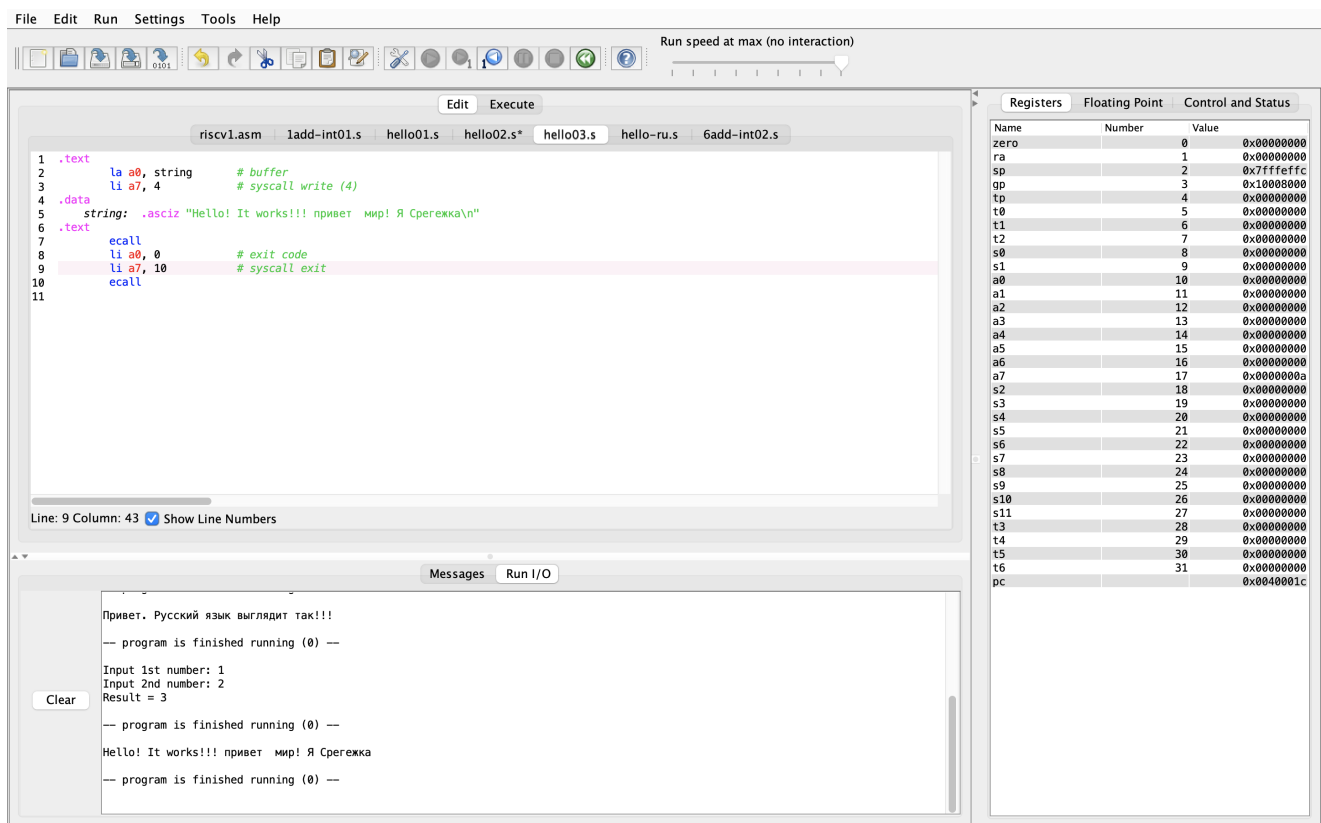
Псевдокоманды в программах:

1. `mv` : Здесь `mv` является псевдокомандой, так как она заменяет собой две.
2. `la` : Это также псевдокоманда, которая используется для загрузки адреса в регистр.

В программе №6 используются команды следующих форматов:

- **R-формат (Register format)**: используется для арифметических и логических команд, где операнды и результат находятся в регистрах. `add mv`
- **I-формат (Immediate format)**: используется для команд, где один из операндов указывается непосредственно в инструкции. `la li`

Запуск всех программ



изменил текстовый вывод

hello.01 и hello.ru отличаются в одну строчку русский и английский выводятся идентично.

```
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: 10 Column: 1 ☒ Show Line Numbers

Messages

Run I/O

Hello! It works!!!

-- program is finished running (0) --

Clear

riscv1(мое первое слово).asm | 1add-int01.s | hello01.s | hello02.s | hello03.s | hello-ru.s

```
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
```

Line: 8 Column: 6 ☒ Show Line Numbers

Messages

Run I/O

Hello! It works!!!

-- program is finished running (0) --

Привет. Русский язык выглядит так!!!

-- program is finished running (0) --

Clear

Первая программа

riscv1(мое первое слово).asm 1add-int01.s hello01.s hello02.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 – останов программы
9      ecall
10 |
```

Line: 10 Column: 1 ☒ Show Line Numbers

Messages Run I/O

4
5
9
-- program is finished running (0) --

Clear

Третья программа

riscv1(мое первое слово).asm

1add-int01.s

hello01.s

hello02.s

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

Line: 9 Column: 1 ☒ Show Line Numbers

Messages

Run I/O

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

Clear

Четвертая программа

riscv1(мое первое слово).asm | 1add-int01.s | hello01.s | hello02.s | hello03.s

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

Line: 11 Column: 1 ☒ Show Line Numbers

Messages Run I/O

Hello! It works!!!
-- program is finished running (0) --
Clear

Шестая программа

```
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
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
```

Line: 29 Column: 49 ☒ Show Line Numbers

Messages

Run I/O

Input 1st number: 52
Input 2nd number: 812
Result = 864

-- program is finished running (0) --

Clear