

Лабораторная работа 6

Мазуркевич Анастасия Дмитриевна

Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	7
Выводы	19
Список литературы	20

Список иллюстраций

1	создаем каталог	7
2	вводим	8
3	создаем и запускаем	8
4	заменяем строки	9
5	проверяем работу файла	9
6	создаем	9
7	вводим код из листинга	10
8	запускаем	10
9	заменяем строки	11
10	создаем и проверяем	12
11	заменяем строки	12
12	проверяем	12
13	создаем	12
14	вводим листинг	13
15	создаем и запускаем, результат верный	13
16	изменяем	14
17	проверяем, все верно	14
18	создаем	14
19	вводим листинг	15
20	работает, 4 вариант	16
21	создаем файл	17
22	код	17
23	все верно	18
24	все верно	18

Список таблиц

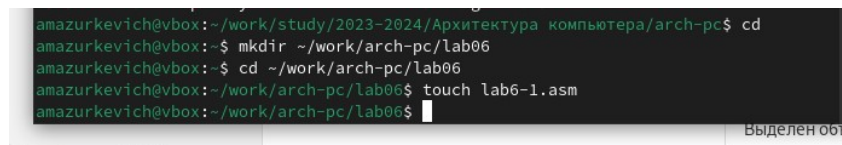
Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

Задание

Выполнение лабораторной работы

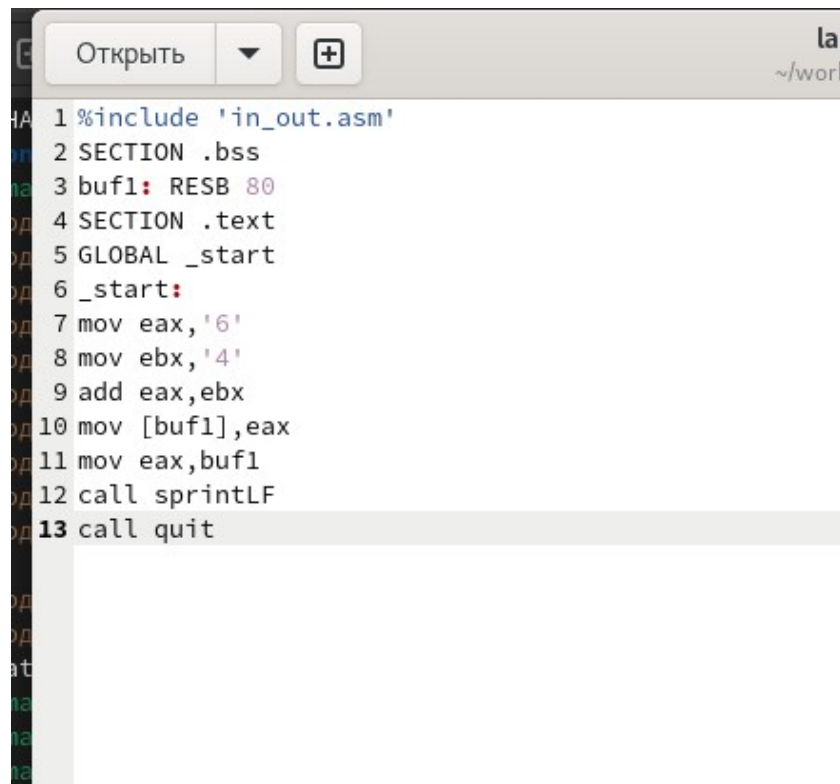
1. Создайте каталог для программ лабораторной работы № 6, перейдите в него и создайте файл lab6-1.asm:



```
amazurkevich@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc$ cd  
amazurkevich@vbox: ~$ mkdir ~/work/arch-pc/lab06  
amazurkevich@vbox: ~$ cd ~/work/arch-pc/lab06  
amazurkevich@vbox: ~/work/arch-pc/lab06$ touch lab6-1.asm  
amazurkevich@vbox: ~/work/arch-pc/lab06$
```

Рис. 1: создаем каталог

Введите в файл lab6-1.asm текст программы из листинга 6.1

A screenshot of a text editor window. The title bar shows buttons for 'Открыть' (Open), a dropdown arrow, and a '+' icon. The file path is partially visible as '~/.work'. The code is in assembly format with line numbers 1 through 13. Line 13 is highlighted. The code includes a macro call, section declarations, a buffer definition, and several instructions for moving, adding, and calling functions.

```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintLF
13 call quit
```

Рис. 2: вводим

Создайте исполняемый файл и запустите его.

A screenshot of a terminal window. The prompt is 'amazurkevich@vbox:~/work/arch-pc/lab06\$'. The user enters three commands: 'nasm -f elf lab6-1.asm', 'ld -m elf_i386 -o lab6-1 lab6-1.o', and './lab6-1'. The output shows the file 'lab6-1' being created and then executed. The prompt changes to 'amazurkevich@vbox:~/work/arch-pc/lab06\$' after each command.

```
amazurkevich@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
amazurkevich@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
amazurkevich@vbox:~/work/arch-pc/lab06$ ./lab6-1
j
amazurkevich@vbox:~/work/arch-pc/lab06$
```

Рис. 3: создаем и запускаем

Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправьте текст программы (Листинг 6.1) следующим образом: замените строки


```

1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 mov [buf1],eax
11 mov eax,buf1
12 call sprintLF
13 call quit

```

Рис. 4: заменяем строки

Создайте исполняемый файл и запустите его. Отображается ли символ? Нет, не отображается

```

amazurkevich@vbox:~/work/arch-pc/lab06$ gedit lab6-1.asm
amazurkevich@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
amazurkevich@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
amazurkevich@vbox:~/work/arch-pc/lab06$ ./lab6-1

amazurkevich@vbox:~/work/arch-pc/lab06$

```

Рис. 5: проверяем работу файла

Создайте файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введите в него текст программы из листинга 6.2.

```

amazurkevich@vbox:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm

```

Рис. 6: создаем

```

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,'6'
6 mov ebx,'4'
7 add eax,ebx
8 call iprintLF
9 call quit

```

Рис. 7: вводим код из листинга

Создайте исполняемый файл и запустите его.

```

amazurkevich@vbox:~/work/arch-pc/lab06$ gedit lab6-2.asm
amazurkevich@vbox:~/work/arch-pc/lab06$ gedit lab6-2.asm
amazurkevich@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
amazurkevich@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
amazurkevich@vbox:~/work/arch-pc/lab06$ ./lab6-2
106
amazurkevich@vbox:~/work/arch-pc/lab06$

```

Листинг 6.2. Программа вывода значения регистра eax

Рис. 8: запускаем

Аналогично предыдущему примеру изменим символы на числа. Замените строки

```
1 %include 'in_out.  
2 SECTION .text  
3 GLOBAL _start  
4 _start:  
5 mov eax,6  
6 mov ebx,4  
7 add eax,ebx  
8 call iprintLF  
9 call quit
```

Рис. 9: заменяем строки

Создайте исполняемый файл и запустите его. Какой результат будет получен при исполнении программы? Результат - 10

```

amazurkevich@vbox: ~/work/arch-pc/lab06$ gedit lab6-2.asm
amazurkevich@vbox: ~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
amazurkevich@vbox: ~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
amazurkevich@vbox: ~/work/arch-pc/lab06$ ./lab6-2
10

```

Рис. 10: создаем и проверяем

Замените функцию `iprintLF` на `iprint`. Создайте исполняемый файл и запустите его. Чем отличается вывод функций `iprintLF` и `iprint`? Разница в переносе строки

```

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprint
9 call quit

```

Рис. 11: заменяем строки

```

amazurkevich@vbox: ~/work/arch-pc/lab06$ gedit lab6-2.asm
amazurkevich@vbox: ~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
amazurkevich@vbox: ~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
amazurkevich@vbox: ~/work/arch-pc/lab06$ ./lab6-2
10amazurkevich@vbox: ~/work/arch-pc/lab06$

```

Рис. 12: проверяем

Создайте файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06`

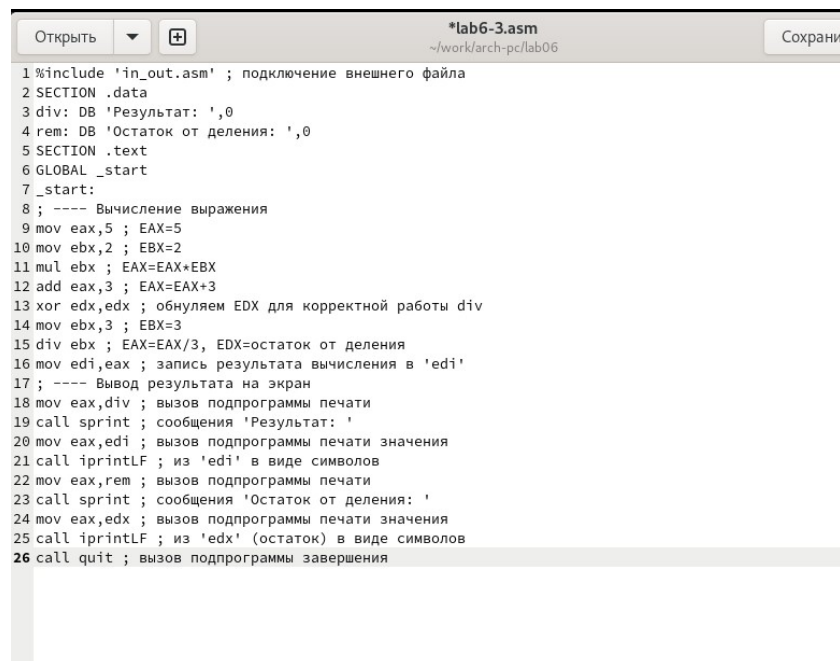
```

10amazurkevich@vbox: ~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
amazurkevich@vbox: ~/work/arch-pc/lab06$ gedit lab6-3.asm
amazurkevich@vbox: ~/work/arch-pc/lab06$

```

Рис. 13: создаем

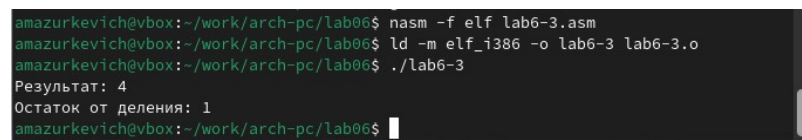
Внимательно изучите текст программы из листинга 6.3 и введите в lab6-3.asm.



```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,5 ; EAX=5
10 mov ebx,2 ; EBX=2
11 mul ebx ; EAX=EAX*EBX
12 add eax,3 ; EAX=EAX+3
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,3 ; EBX=3
15 div ebx ; EAX=EAX/3, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения
```

Рис. 14: вводим листинг

Создайте исполняемый файл и запустите его.



```
amazurkevich@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
amazurkevich@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
amazurkevich@vbox:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
amazurkevich@vbox:~/work/arch-pc/lab06$
```

Рис. 15: создаем и запускаем, результат верный

Измените текст программы для вычисления выражения $\lfloor (4 \cdot 6 + 2) / 5 \rfloor$

```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,4 ; EAX=4
10 mov ebx,6 ; EBX=6
11 mul ebx ; EAX=EAX*EBX
12 add eax,2 ; EAX=EAX+2
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,5 ; EBX=5
15 div ebx ; EAX=EAX/5, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения

```

Загрузка файла «~/work/arch-pc/lab06/lab6-3.asm»... Matlab Ширина табуляции: 8 Ln 15, Col 1

Рис. 16: изменяем

Создайте исполняемый файл и проверьте его работу

```

amazurkevich@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
amazurkevich@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
amazurkevich@vbox:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
amazurkevich@vbox:~/work/arch-pc/lab06$

```

Рис. 17: проверяем, все верно

Создайте файл variant.asm в каталоге ~/work/arch-pc/lab06:

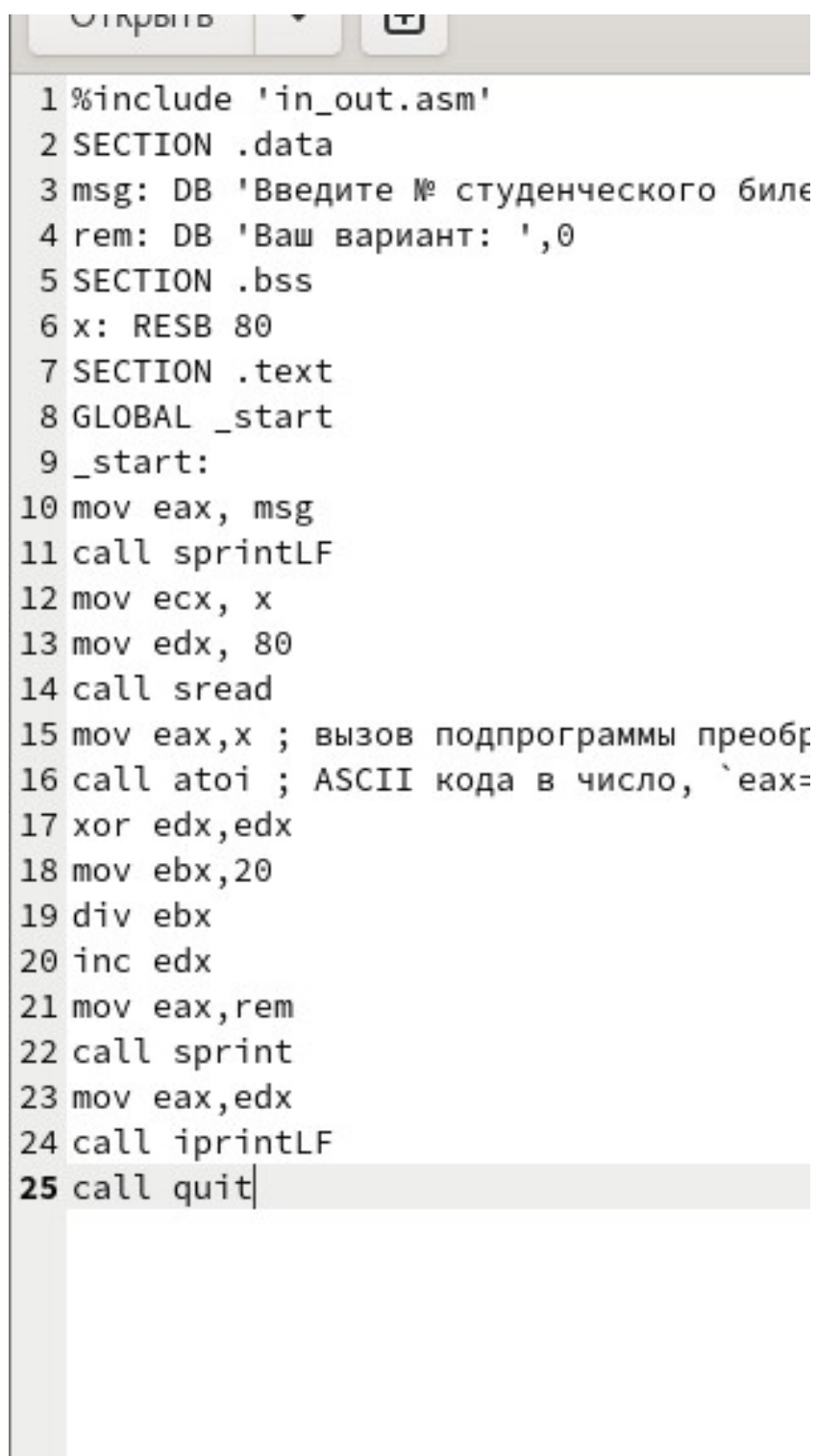
```

amazurkevich@vbox:~/work/arch-pc/lab06$ gedit lab6-3.asm
amazurkevich@vbox:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm

```

Рис. 18: создаем

Внимательно изучите текст программы из листинга 6.4 и введите в файл variant.asm.



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите № студенческого биле
4 rem: DB 'Ваш вариант: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax,x ; вызов подпрограммы преобр
16 call atoi ; ASCII кода в число, `eax=
17 xor edx,edx
18 mov ebx,20
19 div ebx
20 inc edx
21 mov eax,rem
22 call sprintf
23 mov eax,edx
24 call iprintLF
25 call quit
```

Рис. 19: вводим листинг

Проверьте работу файла, получите ваш вариант

```
amazurkevich@vbox: ~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246843
Ваш вариант: 4
amazurkevich@vbox: ~/work/arch-pc/lab06$
```

Рис. 20: работает, 4 вариант

Включите в отчет по выполнению лабораторной работы ответы на следующие вопросы: 1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? `mov eax,rem call sprint` 2. Для чего используются следующие инструкции? `mov ecx, x mov edx, 80 call sread` Для чтения строки с данными, которые вводит пользователь

3. Для чего используется инструкция “`call atoi`”? Используется для преобразования строки в целое число. Принимает адрес строки в регистре `eax` и возвращает полученное число в регистре `eax`.
4. Какие строки листинга 6.4 отвечают за вычисления варианта? Строка “`xor edx,edx`” обнуляет регистр `edx` перед делением. “`mov ebx,20`” загружает значение 20 в `ebx`. “`div ebx`” деление `eax` на значение регистра `ebx` с сохранением частного в регистре `eax` и остатка в регистре `edx`.
5. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”? Остаток от деления записывается в регистр `edx`.
6. Для чего используется инструкция “`inc edx`”? Для увеличения значения в регистре `edx` на 1. В данном случае увеличивает остаток от деления на 1
7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений? “`mov eax,edx`” передает значение остатка от деления в регистр `eax`. “`call iprintLF`” вызывает процедуру `iprintLF` для вывода значения на экран вместе с переводом строки.

САМОСТОЯТЕЛЬНАЯ РАБОТА Написать программу вычисления выражения $\pi = \pi(\pi)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения π , вычислять заданное выражение в зависимости от введенного π , выводить результат вычислений. Вид функции $\pi(\pi)$ выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений π_1 и π_2 из 6.3.

Для начала создадим новый файл

```

bash вариант: 4
amazurkevich@vbox:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-4.asm
amazurkevich@vbox:~/work/arch-pc/lab06$

```

Рис. 21: создаем файл

Пишем программу для вычисления функции

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 div: DB 'Результат: ',0
5 SECTION .bss
6 rez: RESB 80
7 x: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 mov eax,msg
12 call sprintf
13
14 mov ecx,x
15 mov edx,80
16 call sread
17 mov eax,x
18 call atoi
19
20 sub eax,1
21 mov ebx,4
22 imul eax,ebx
23 mov ebx,3
24 xor edx,edx
25 idiv ebx
26 add eax,5
27 mov [rez],eax
28
29 mov eax,div
30 call sprintf
31 mov eax,[rez]
32 call iprintLF
33 call quit

```

Рис. 22: код

Проверяем для $x=4$

```
amazurkevich@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите x:
4
Результат: 9
amazurkevich@vbox:~/work/arch-pc/lab06$
```

Рис. 23: все верно

Проверяем для $x=10$

```
amazurkevich@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
amazurkevich@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите x:
10
Результат: 17
amazurkevich@vbox:~/work/arch-pc/lab06$
```

Рис. 24: все верно

Выводы

Освоили арифметические инструкции языка ассемблера NASM.

Список литературы