

# **Отчёт по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

Тимофеева Екатерина Николаевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Ответы на вопросы по программе</b>	<b>12</b>
<b>5</b>	<b>Выполнение заданий для самостоятельной работы</b>	<b>13</b>
<b>6</b>	<b>Выводы</b>	<b>15</b>
	<b>Список литературы</b>	<b>16</b>

## Список иллюстраций

3.1	Создание директории и переход в каталог . . . . .	8
3.2	Создание файла . . . . .	8
3.3	Ввод текста . . . . .	8
3.4	Создание файла . . . . .	9
3.5	Замена текста программы . . . . .	9
3.6	Запуск файла . . . . .	9
3.7	Создание файла . . . . .	9
3.8	Создание и запуск файла . . . . .	10
3.9	Создание и запуск нового файла . . . . .	10
3.10	Создание файла . . . . .	10
3.11	Создание и запуск файла . . . . .	10
3.12	Создание и запуск нового файла . . . . .	11
3.13	Создание файла . . . . .	11
3.14	Создание и запуск файла, ввод студенческого билета . . . . .	11
5.1	Создание файла . . . . .	13
5.2	Создание и запуск файла, проверка . . . . .	13
5.3	Программа . . . . .	14

## **Список таблиц**

# 1 Цель работы

Цель работы заключается в освоении арифметических инструкций языка ассемблера NASM.

## 2 Задание

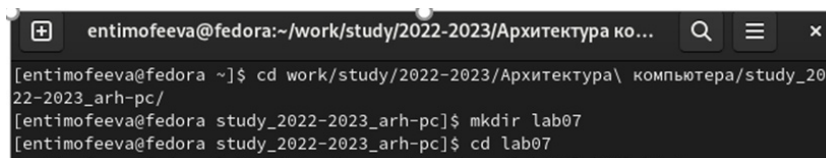
1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

### 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно. # Выполнение лабораторной работы

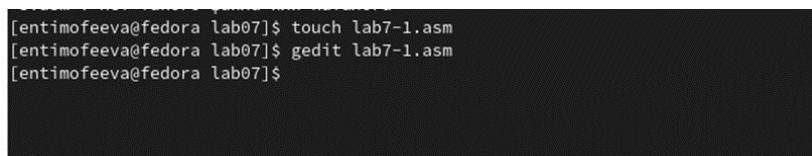
№1. Создаём директорию, в которой будем создавать файлы с программами для лабораторной работы №7 и переходим в созданный каталог. (рис. 3.1)



```
entimofeeva@fedora:~/work/study/2022-2023/Архитектура ко...
[entimofeeva@fedora ~]$ cd work/study/2022-2023/Архитектура\ компьютера/study_20
22-2023_arh-pc/
[entimofeeva@fedora study_2022-2023_arh-pc]$ mkdir lab07
[entimofeeva@fedora study_2022-2023_arh-pc]$ cd lab07
```

Рис. 3.1: Создание директории и переход в каталог

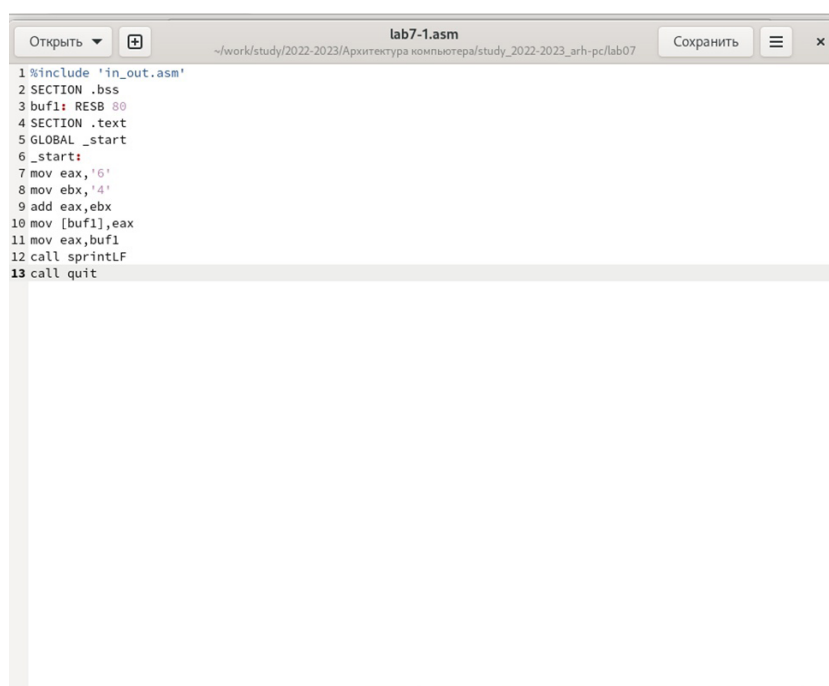
Создаём файл lab7-1.asm и открываем его. (рис. 3.2)



```
[entimofeeva@fedora lab07]$ touch lab7-1.asm
[entimofeeva@fedora lab07]$ gedit lab7-1.asm
[entimofeeva@fedora lab07]$
```

Рис. 3.2: Создание файла

Вводим в файл текст программы из листинга. (рис. 3.3)



```
lab7-1.asm
~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab07
Сохранить

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 sprintf
13 call quit
```

Рис. 3.3: Ввод текста

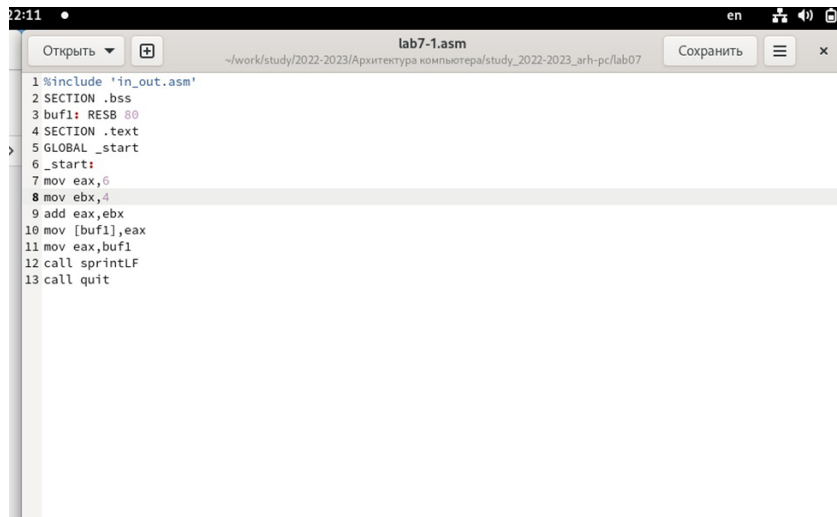
Создаём исполняемый файл программы и запускаем, программа вывела символ j. (рис. 3.4)



```
[entimofeeva@fedora lab07]$ nasm -f elf lab7-1.asm
[entimofeeva@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[entimofeeva@fedora lab07]$ ./lab7-1
j
```

Рис. 3.4: Создание файла

Далее изменяем текст программы, вместо стмволов “6” и “4” вводим 6 и 4. (рис. 3.5)



```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax,0
8 mov ebx,4
9 add eax,ebx
10 mov [buf1],eax
11 mov eax,buf1
12 call sprintf
13 call quit
```

Рис. 3.5: Замена текста программы

Создаём новый исполняемый файл программы и запускаем его. Вывелся символ с кодом 10, это символ перевода строки и он не отображается при выводе на экран. (рис. 3.6)

```
[entimofeeva@fedora lab07]$ gedit lab7-1.asm
[entimofeeva@fedora lab07]$ nasm -f elf lab7-1.asm
[entimofeeva@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[entimofeeva@fedora lab07]$ ./lab7-1

[entimofeeva@fedora lab07]$
```

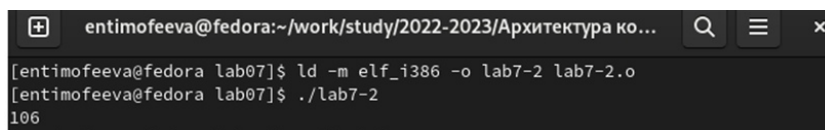
Рис. 3.6: Запуск файла

Создаём новый файл lab7-2.asm (рис. 3.7)

```
entimofeeva@fedora lab07]$ touch lab7-2.asm
entimofeeva@fedora lab07]$ gedit lab7-2.asm
entimofeeva@fedora lab07]$ nasm -f elf lab7-2.asm
```

Рис. 3.7: Создание файла

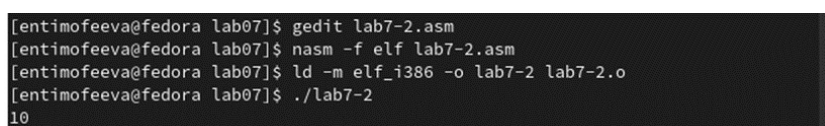
Вводим в файл текст другой программы, создаём и запускаем исполняемый файл, теперь вывело число 106. (рис. 3.8)



```
entimofeeva@fedora:~/work/study/2022-2023/Архитектура ко...
[entimofeeva@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[entimofeeva@fedora lab07]$ ./lab7-2
106
```

Рис. 3.8: Создание и запуск файла

Заменяем в тексте программы символы, создаём и запускаем новый исполняемый файл, программа вывела 10. (рис.3.9)

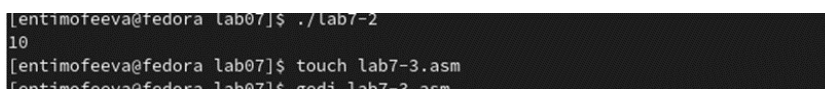


```
[entimofeeva@fedora lab07]$ gedit lab7-2.asm
[entimofeeva@fedora lab07]$ nasm -f elf lab7-2.asm
[entimofeeva@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[entimofeeva@fedora lab07]$ ./lab7-2
10
```

Рис. 3.9: Создание и запуск нового файла

Заменяем в тексте программы функцию `iprintLF` на `iprint` и запускаем новый файл, вывод не изменился.

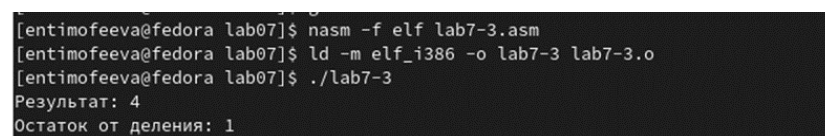
№2. Создаём файл `lab7-3.asm` (рис. 3.10)



```
[entimofeeva@fedora lab07]$ ./lab7-2
10
[entimofeeva@fedora lab07]$ touch lab7-3.asm
[entimofeeva@fedora lab07]$ gedit lab7-3.asm
```

Рис. 3.10: Создание файла

Вводим в созданный файл текст программы для вычисления значения выражения, создаём исполняемый файл и запускаем его. (рис. 3.11)



```
[entimofeeva@fedora lab07]$ nasm -f elf lab7-3.asm
[entimofeeva@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[entimofeeva@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
```

Рис. 3.11: Создание и запуск файла

Изменяем программу, создаём и запускаем новый исполняемый файл. (рис. 3.12)

```

[entimofeeva@fedora lab07]$ gedit lab7-3.asm
[entimofeeva@fedora lab07]$ nasm -f elf lab7-3.asm
[entimofeeva@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[entimofeeva@fedora lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1

```

Рис. 3.12: Создание и запуск нового файла

Создаём файл variant.asm (рис. 3.13)

```

entimofeeva@fedora:~/work/study/2022-2023/Архитектура ко...
[entimofeeva@fedora lab07]$ touch variant.asm

```

Рис. 3.13: Создание файла

Вводим в файл текст программы для вычисления варианта задания по номеру студенческого билета, создаём и запускаем исполняемый файл, вводим номер своего студенческого билета, вывело 7 вариант. (рис. 3.14)

```

[entimofeeva@fedora lab07]$ nasm -f elf variant.asm
[entimofeeva@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[entimofeeva@fedora lab07]$ ./variant
Введите No студенческого билета:
1132226446
Ваш вариант: 7

```

Рис. 3.14: Создние и запуск файла, ввод студенческого билета

## 4 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```
mov eax,rem call sprint
```

2. Инструкция `mov ecx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx` `mov edx, 80` - запись в регистр `edx` длины вводимой строки `call sread` - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры.

3. `call atoi` используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`.

4. За вычисления варианта отвечают строки:

```
xor edx,edx ; обнуление edx для корректной работы div
mov ebx,20 ; ebx = 20
div ebx ; eax = eax/20, edx - остаток от деления
inc edx ; edx = edx + 1
```

5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`.

6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1

7. За вывод на экран результатов вычислений отвечают строки:

```
mov eax,edx call iprintLF
```

## 5 Выполнение заданий для самостоятельной работы

Создаём файл для написания программы. (рис. 5.1)

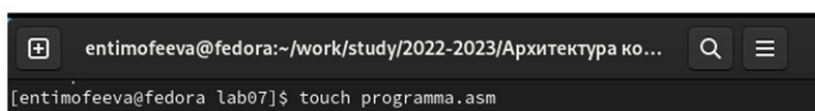


Рис. 5.1: Создание файла

Открываем созданный файл, вводим в него текст программы для вычисления выражения, создаём и запускаем исполняемый файл, выполняем проверки. (рис. 5.2)

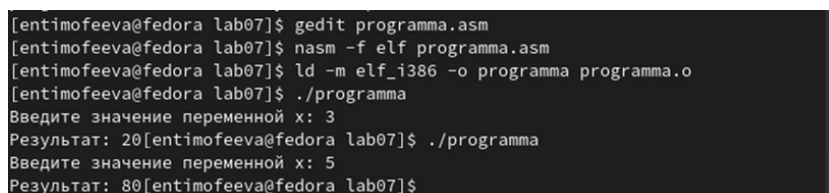


Рис. 5.2: Создание и запуск файла, проверка

Программа для вычисления выражения  $5 \cdot (x - 1)^2$  (рис. 5.3)

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg: DB 'Введите значение переменной x: ',0
4 rem: DB 'Результат: ',0
5 SECTION .bss
6 x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры
7 SECTION .text
8 GLOBAL _start
9 _start:
10 ; — Вычисление выражения
11 mov eax, msg
12 call sprint
13 mov ecx, x
14 mov edx, 80
15 call sread
16 mov eax, x ; вызов подпрограммы преобразования
17 call atoi ; ASCII кода в число, eax=x
18 sub eax, 1
19 mul eax
20 mov ebx, 5
21 mul ebx
22 mov edi, eax ; запись результата вычисления в 'edi'
23 ; — Вывод результата на экран
24 mov eax, rem ; вызов подпрограммы печати
25 call sprint ; сообщения 'Результат: '
26 mov eax, edi ; вызов подпрограммы печати значения
27 call iprint ; из 'edi' в виде символов
28 call quit ; вызов подпрограммы завершения
```

Рис. 5.3: Программа

## 6 Выводы

Мы освоили арифметические инструкции языка ассемблера NASM.

# Список литературы

1. Лабораторная работа №7.
2. Таблица ASCII