

# **Шаблон отчёта по лабораторной работе**

**7**

Сильвен Макс Грегор Филс , НКАбд-03-22

# Содержание

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Цель работы</b>                                    | <b>5</b>  |
| <b>2</b> | <b>Выполнение лабораторной работы :</b>               | <b>6</b>  |
| 2.1      | Символьные и численные данные в NASM : . . . . .      | 6         |
| 2.2      | Выполнение арифметических операций в NASM : . . . . . | 11        |
| 2.3      | Вопросы : . . . . .                                   | 16        |
| 2.4      | Выводы по результатам выполнения заданий : . . . . .  | 16        |
| <b>3</b> | <b>Задание для самостоятельной работы :</b>           | <b>17</b> |
| 3.1      | Выводы по результатам выполнения заданий : . . . . .  | 19        |
| <b>4</b> | <b>Выводы</b>   | <b>20</b> |

## Список иллюстраций

|      |                      |    |
|------|----------------------|----|
| 2.1  | Ресунок 1 . . . . .  | 6  |
| 2.2  | Ресунок 2 . . . . .  | 7  |
| 2.3  | Ресунок 3 . . . . .  | 7  |
| 2.4  | Ресунок 4 . . . . .  | 8  |
| 2.5  | Ресунок 5 . . . . .  | 8  |
| 2.6  | Ресунок 6 . . . . .  | 9  |
| 2.7  | Ресунок 7 . . . . .  | 9  |
| 2.8  | Ресунок 8 . . . . .  | 9  |
| 2.9  | Ресунок 9 . . . . .  | 10 |
| 2.10 | Ресунок 10 . . . . . | 10 |
| 2.11 | Ресунок 13 . . . . . | 12 |
| 2.12 | Ресунок 14 . . . . . | 12 |
| 2.13 | Ресунок 15 . . . . . | 13 |
| 2.14 | Ресунок 16 . . . . . | 13 |
| 2.15 | Ресунок 17 . . . . . | 14 |
| 2.16 | Ресунок 18 . . . . . | 14 |
| 2.17 | Ресунок 19 . . . . . | 15 |
| 2.18 | Ресунок 20 . . . . . | 15 |
| 3.1  | Ресунок 21 . . . . . | 17 |
| 3.2  | Ресунок 22 . . . . . | 18 |

## **Список таблиц**

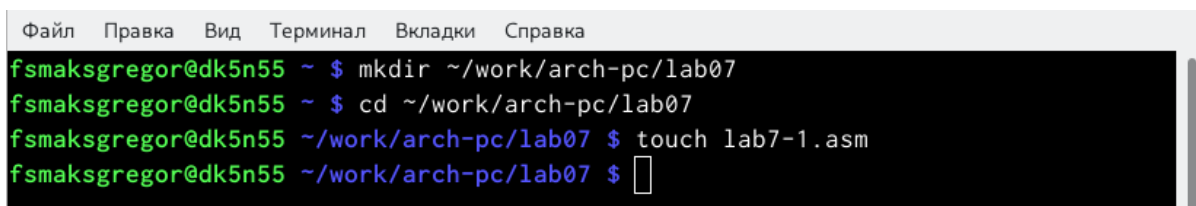
# 1 Цель работы

- В седьмой лабораторной работе можно будет освоить арифметические операции языка ассемблера.

## 2 Выполнение лабораторной работы :

### 2.1 Символьные и численные данные в NASM :

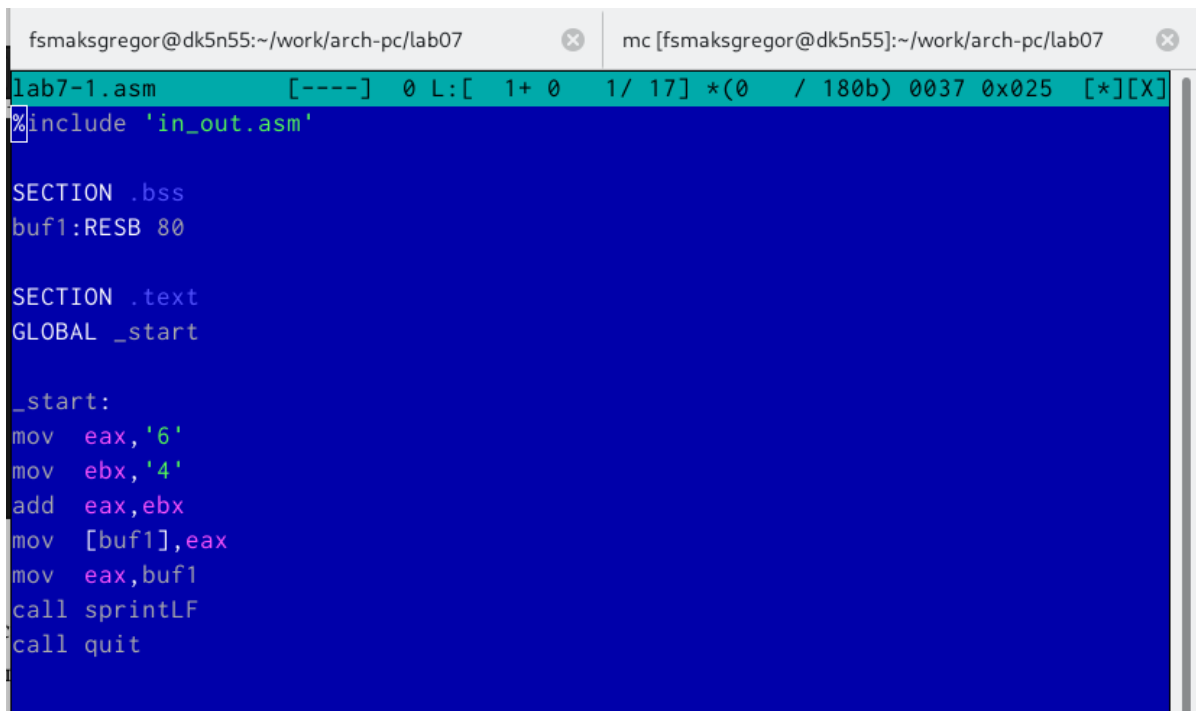
- Здесь мы начали с создания, а затем переместились в седьмой каталог ла- боратории “~/work/arch-pc/lab07”, после чего мы создали файл “lab7-1.asm”.(рис. ??)



```
Файл  Правка  Вид  Терминал  Вкладки  Справка
fsmaksgregor@dk5n55 ~ $ mkdir ~/work/arch-pc/lab07
fsmaksgregor@dk5n55 ~ $ cd ~/work/arch-pc/lab07
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ touch lab7-1.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

Рис. 2.1: Ресунок 1

- После этого мы заполнили файл .asm кодом программы, отображающей значение регистра eax. (рис. ??)



```
lab7-1.asm      [----]  0 L:[  1+ 0  1/ 17] *(0  / 180b) 0037 0x025  [*][X]
%include 'in_out.asm'

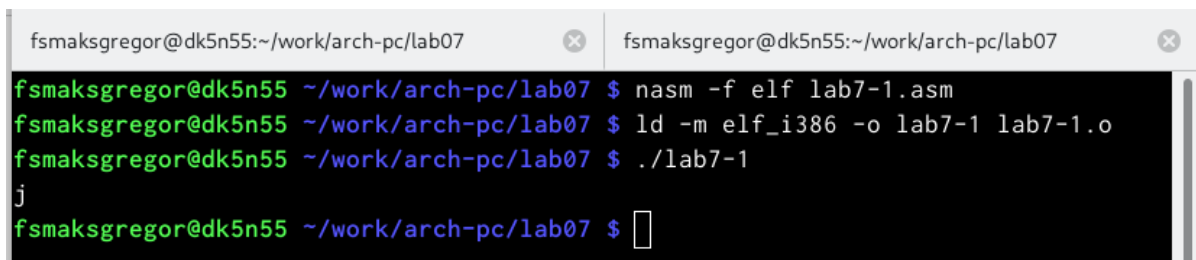
SECTION .bss
buf1:RESB 80

SECTION .text
GLOBAL _start

_start:
mov  eax,'6'
mov  ebx,'4'
add  eax,ebx
mov  [buf1],eax
mov  eax,buf1
call sprintf
call quit
```

Рис. 2.2: Ресунок 2

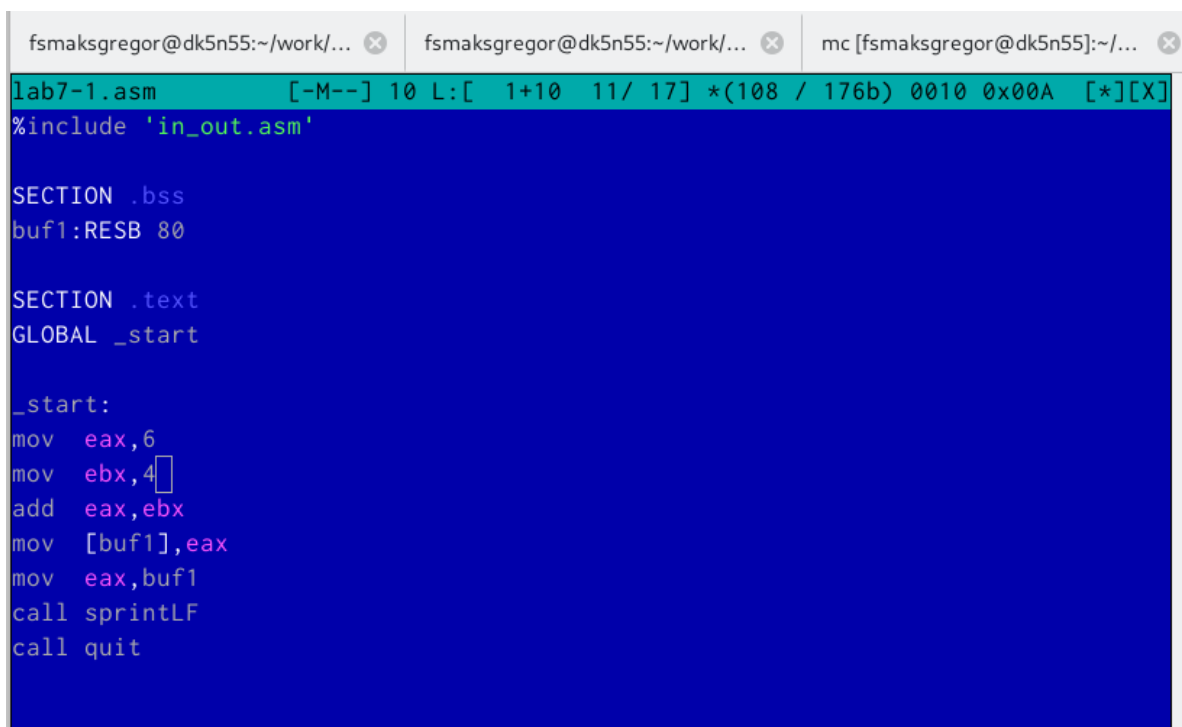
- Затем мы скомпилировали файл, создали исполняемый файл и запустили программу, все это после перемещения файла in\_out.asm в тот же каталог, где находится lab7-1.asm.(рис. ??)



```
fsmaksgregor@dk5n55:~/work/arch-pc/lab07
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ./lab7-1
j
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

Рис. 2.3: Ресунок 3

- После этого мы изменили код в листинге следующим образом : mov eax,6  
mov ebx,4 (рис. ??)



```
lab7-1.asm      [-M--] 10 L:[ 1+10 11/ 17] *(108 / 176b) 0010 0x00A  [*][X]
#include 'in_out.asm'

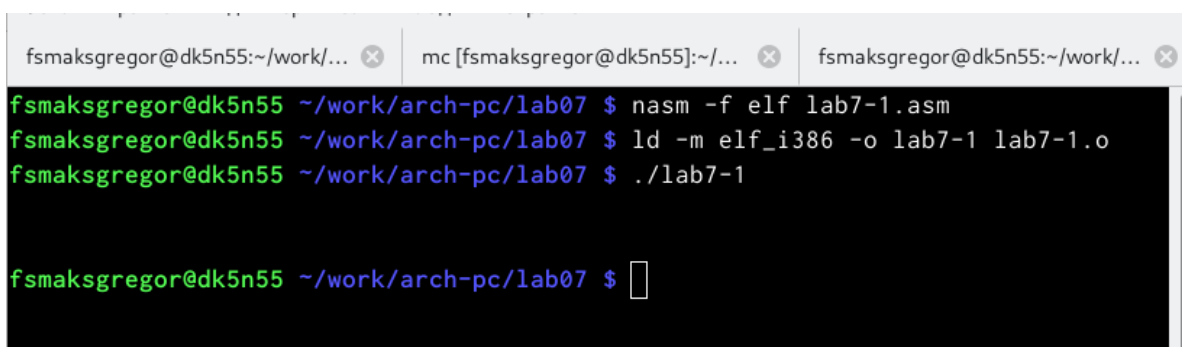
SECTION .bss
buf1:RESB 80

SECTION .text
GLOBAL _start

_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
mov  [buf1],eax
mov  eax,buf1
call printf
call _exit
```

Рис. 2.4: Ресунок 4

- Затем мы снова скомпилировали файл и создали исполняемый файл.(рис. ??)



```
fsmaksgregor@dk5n55:~/work/...  mc [fsmaksgregor@dk5n55]:~/...  fsmaksgregor@dk5n55:~/work/...
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ./lab7-1

fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

Рис. 2.5: Ресунок 5

- Проверив ASCII table символ, соответствующий коду 10 это новая строка, и мы можем сказать, что это было отображено, потому что при запуске программы она отобразила новую строку в качестве вывода.



- После этого мы создали файл lab-2.asm, в котором мы использовали подпрограммы, расположенные в файле in\_out.asm. (рис. ??)

```
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ touch ~/work/arch-pc/lab07/lab7-2.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

Рис. 2.6: Ресунок 6

- После этого мы заполнили файл необходимым кодом для вывода значения реестра с помощью подпрограммы. (рис. ??)

```
lab7-2.asm [----] 9 L: [ 1+12 13/ 13] *(130 / 130b) <EOF> [*][X]
#include 'in_out.asm'

SECTION .text
GLOBAL _start

_start:

mov     eax, '6'
mov     ebx, '4'
add     eax, ebx
call    iprintLF

call    quit
```

Рис. 2.7: Ресунок 7

- мы скомпилировали файл, создали исполняемый файл и запустили его. (рис. ??)

```
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ./lab7-2
106
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

Рис. 2.8: Ресунок 8

- Аналогично предыдущему примеру, мы меняем символы на цифры, заменяя строки на : `mov eax,6 mov ebx,4` (рис. ??)

```
lab7-2.asm [----] 9 L:[ 1+12 13/ 13] *(126 / 126b) <EOF> [*][X]
#include 'in_out.asm'

SECTION .text
GLOBAL _start

_start:

mov    eax,6
mov    ebx,4
add    eax,ebx
call   iprintLF

call   quit
```

Рис. 2.9: Ресунок 9

- Затем мы снова скомпилировали файл и создали исполняемый файл.(рис. ??)

```
fsmaksgregor@dk5n55:~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
fsmaksgregor@dk5n55:~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
fsmaksgregor@dk5n55:~/work/arch-pc/lab07 $ ./lab7-2
10
fsmaksgregor@dk5n55:~/work/arch-pc/lab07 $
```

Рис. 2.10: Ресунок 10

- На этот раз результатом, который мы получили, действительно было добавление 6 и 4 который 10.

-Затем мы заменили функцию `iprintLF` на `iprint`. После этого был создан исполняемый файл, и мы запустили его. (рис. ??) (рис. ??)

The screenshot shows a terminal window with two panes. The top pane displays the assembly code for `lab7-2.asm`. The code includes a header, a section declaration, a global symbol, and assembly instructions. The bottom pane shows the execution of the assembly code using `nasm`, `ld`, and `./lab7-2`.

```
lab7-2.asm [-M--] 9 L: [ 1+12 13/ 13] *(124 / 124b) <EOF> [*][X]
#include 'in_out.asm'

SECTION .text
GLOBAL _start

_start:

mov     eax,6
mov     ebx,4
add     eax,ebx
call    iprint

call    quit
```

```
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ./lab7-2
10fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

- Когда мы использовали подпрограмму `iprint`, мы заметили, что вывод отличается от предыдущего, потому что при использовании `iprint` не создается новая строка после вывода.

## 2.2 Выполнение арифметических операций в NASM :

- В качестве примера выполнения арифметических операций в NASM введем программу вычисления арифметического выражения  $\boxed{x}(\boxed{x}) = (5 \boxed{x} 2 + 3)/3$
- Мы создали файл `lab7-3.asm` в каталоге `~/work/arch-pc/lab07`. (рис. ??)

```
mc [fsmaksgregor@dk5n55]:~/work/arch-pc/lab07 x fsmaksgregor@dk5n55:~/work/arch-pc/lab07 x
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ touch lab7-3.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.o lab7-3.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

Рис. 2.11: Ресунок 13

- Затем мы заполнили файл необходимым кодом.(рис. ??)

```
fsmaksgregor@dk5n55:~/work/arch-pc/lab07 x mc [fsmaksgregor@dk5n55]:~/work/arch-pc/lab07 x
lab7-3.asm [----] 0 L: [ 1+10 11/ 33] *(156 / 351b) 0010 0x00A [*][X]
#include 'in_out.asm

SECTION .data

div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0

SECTION .text
GLOBAL _start
_start:
mov eax,5
mov ebx,2
mul ebx
add eax,3
xor edx,edx
mov ebx,3
div ebx

mov edi,eax
mov eax,div
```

Рис. 2.12: Ресунок 14

- Создали исполняемый файл и запустили его.(рис. ??)

```
fsmaksgregor@dk5n55:~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
lab7-3.asm:1: warning: unterminated string [-w+other]
fsmaksgregor@dk5n55:~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
fsmaksgregor@dk5n55:~/work/arch-pc/lab07 $ ./lab7-3
Результат: 4
Остаток от деления: 1
fsmaksgregor@dk5n55:~/work/arch-pc/lab07 $
```

Рис. 2.13: Ресунок 15

- Затем мы изменили текст программы, чтобы вычислить выражение:  $(4 \times 6 + 2) / 5$  (рис. ??)

```
lab7-3.asm [----] 9 L:[ 11+17 28/ 32] *(312 / 351b) 0110 0x06E [*][X]
mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx

mov edi,eax

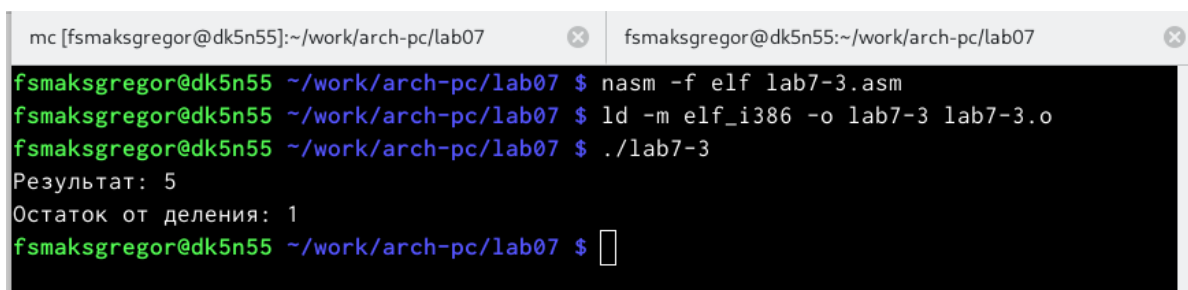
mov eax,div
call sprint
mov eax,edi
call iprintLF

mov eax,rem
call sprint
mov eax,edx
call iprintLF

call quit
```

Рис. 2.14: Ресунок 16

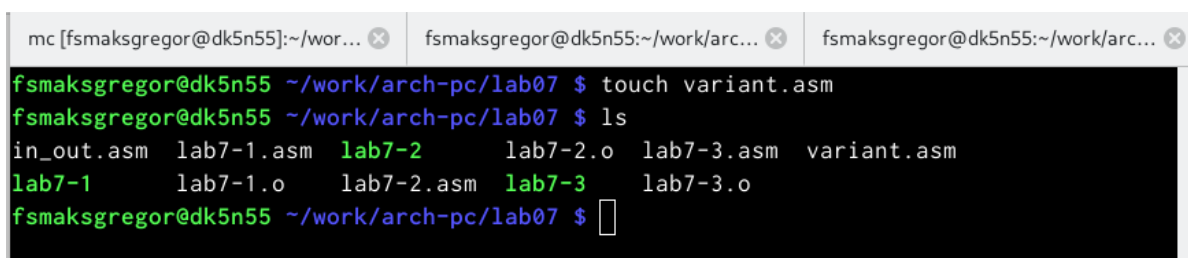
- мы создали исполняемый файл и проверили его работу. (рис. ??)



```
mc [fsmaksgregor@dk5n55]:~/work/arch-pc/lab07 x fsmaksgregor@dk5n55:~/work/arch-pc/lab07 x
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ./lab7-3
Результат: 5
Остаток от деления: 1
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

Рис. 2.15: Ресунок 17

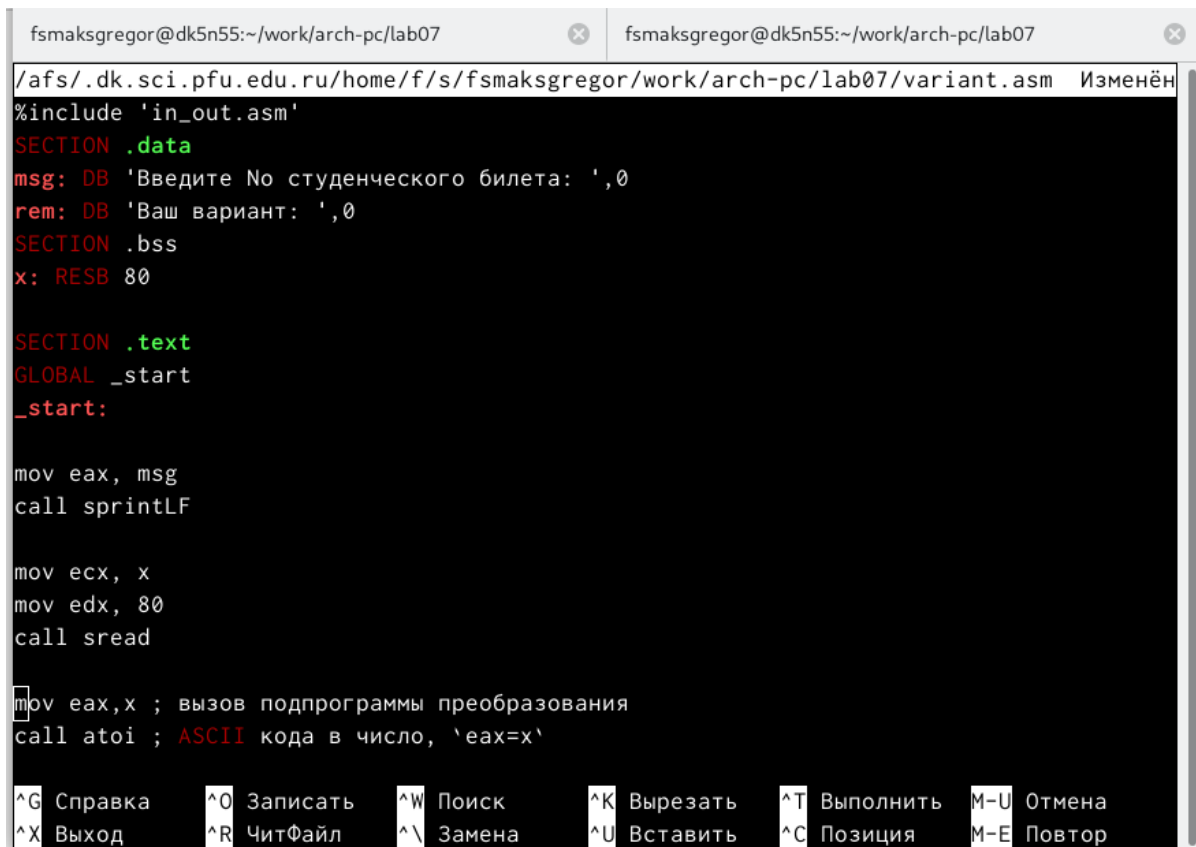
- На этом шаге мы написали программу, которая может вычислить дисперсию, которую мы получаем из номера студенческого билета.
- Мы начали с создания файла `variant.asm`. (рис. ??)



```
mc [fsmaksgregor@dk5n55]:~/wor... x fsmaksgregor@dk5n55:~/work/arc... x fsmaksgregor@dk5n55:~/work/arc... x
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ touch variant.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ls
in_out.asm lab7-1.asm lab7-2 lab7-2.o lab7-3.asm variant.asm
lab7-1 lab7-1.o lab7-2.asm lab7-3 lab7-3.o
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

Рис. 2.16: Ресунок 18

- После этого мы написали код программы. (рис. ??)



```
/afs/.dk.sci.pfu.edu.ru/home/f/s/fsmaksgregor/work/arch-pc/lab07/variant.asm  Изменён
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите No студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintLF

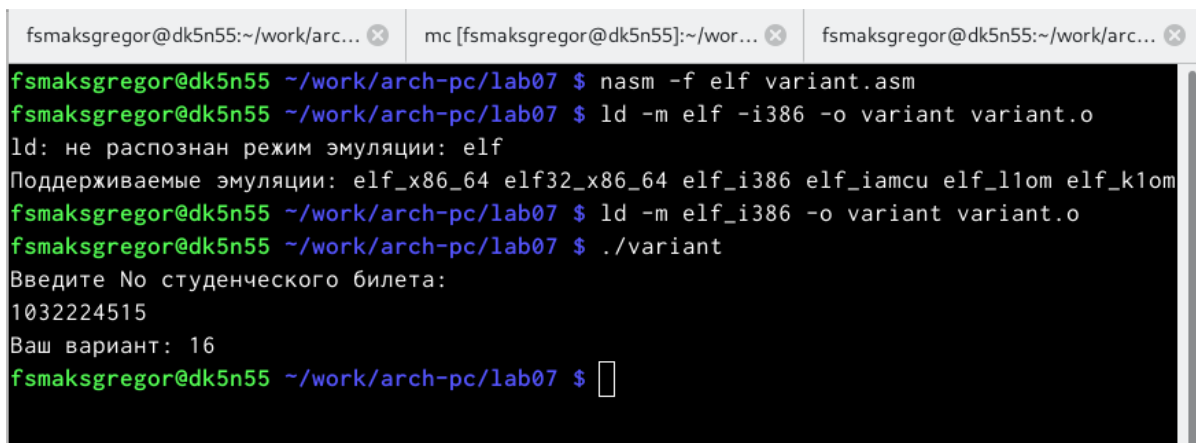
mov ecx, x
mov edx, 80
call sread

mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'

^G Справка    ^O Записать   ^W Поиск      ^K Вырезать   ^T Выполнить  M-U Отмена
^X Выход      ^R ЧитФайл   ^\ Замена     ^U Вставить   ^C Позиция    M-E Повтор
```

Рис. 2.17: Ресунок 19

- мы создали исполняемый файл и проверили его работу, и действительно, в зависимости от номера студента он генерирует номер варианта. (рис. ??)



```
fsmaksgregor@dk5n55:~/work/arch-...  mc [fsmaksgregor@dk5n55]:~/wor...  fsmaksgregor@dk5n55:~/work/arch-...
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ nasm -f elf variant.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ld -m elf -i386 -o variant variant.o
ld: не распознан режим эмуляции: elf
Поддерживаемые эмуляции: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu elf_l10m elf_k10m
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o variant variant.o
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ./variant
Введите No студенческого билета:
1032224515
Ваш вариант: 16
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

Рис. 2.18: Ресунок 20

## 2.3 Вопросы :

- Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? **О** : rem: DB ‘Ваш вариант:’,0 mov eax,rem call sprint
- Для чего используются следующие инструкции? mov ecx, x / mov edx, 80 / call sread **О** : Эти инструкции были использованы для того, чтобы позволить пользователю вводить данные.
- Для чего используется инструкция “call atoi”? **О** : Эта инструкция используется для преобразования значения x из ASCII-кода в целое число.
- Какие строки листинга 7.4 отвечают за вычисления варианта? **О** : xor edx,edx  
mov ebx,20 div ebx inc edx
- В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”? **О** : Остаток был записан в регистре **edx**
- Для чего используется инструкция “inc edx”? **О** : Эта инструкция была использована для увеличения значения в регистре **edx**
- Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений? **О** : mov eax,edx call iprintLF

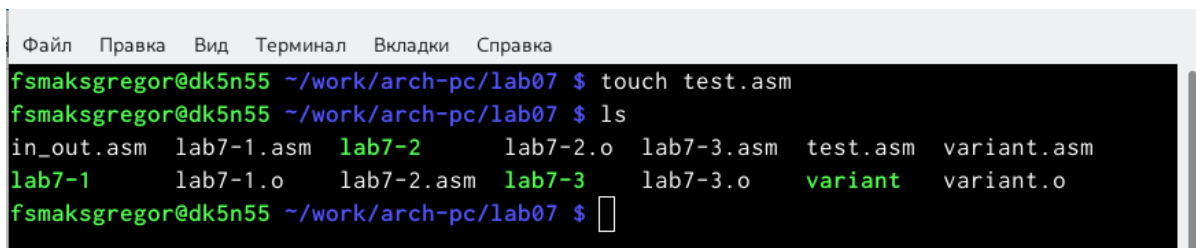
## 2.4 Выводы по результатам выполнения заданий :

- В ходе лабораторной работы мы освоили выполнение арифметических операций на языке ассемблера и углубились в использование подпрограммы.



### 3 Задание для самостоятельной работы :

- В этой работе нам пришлось написать программу, которая просит пользователя ввести значение переменной и решить математическое выражение.
- Мой вариант : 16
- математическое выражение :  $(10x - 5)^2$
- Итак, мы начали с создания asm-файла, в котором будет находиться наш код.(рис. ??)



```
Файл  Правка  Вид  Терминал  Вкладки  Справка
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ touch test.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ls
in_out.asm  lab7-1.asm  lab7-2      lab7-2.o  lab7-3.asm  test.asm  variant.asm
lab7-1      lab7-1.o    lab7-2.asm  lab7-3    lab7-3.o    variant   variant.o
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

Рис. 3.1: Ресунок 21

- После этого мы написали код нашей программы. (рис. ??)

```
терминал - mc [istaksgregol@ukzn33]:~/work/asm-projects/
Файл  Правка  Вид  Терминал  Вкладки  Справка
test.asm  [----]  3 L: [ 9+21  30/ 42] *(392 / 518b) 0032 0x020 [*][X]
x :      RESB 80

SECTION .text
GLOBAL _start
_start:

mov     eax, msg1
call    sprintf

mov     eax, msg
call    sprintf

mov     ecx, x
mov     edx, 80
call    sread

mov     eax, x
call    atoi
mov     edi, eax

mov     eax, 10
mul     edi
sub     eax, 5
mov     edx, eax
mul     edx
mov     edi, eax

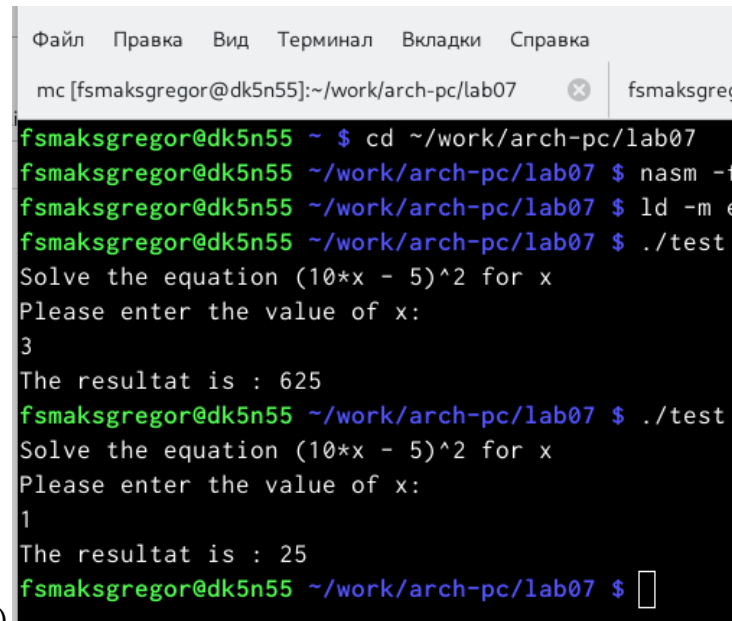
mov     eax, rem
call    sprintf
mov     eax, edi
call    iprintLF

call    quit

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9МенюMC10Выход
```

Рис. 3.2: Рисунок 22

- и, наконец, мы проверяем корректность кода, который мы написали, используя два разных значения  $x_1 = 3$   $x_2 = 1$



```
Файл  Правка  Вид  Терминал  Вкладки  Справка
mc [fsmaksgregor@dk5n55]:~/work/arch-pc/lab07  fsmaksgregor
fsmaksgregor@dk5n55 ~ $ cd ~/work/arch-pc/lab07
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ nasm -f elf32 test.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ld -m elf32 test.o -o test
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ./test
Solve the equation (10*x - 5)^2 for x
Please enter the value of x:
3
The resultat is : 625
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $ ./test
Solve the equation (10*x - 5)^2 for x
Please enter the value of x:
1
The resultat is : 25
fsmaksgregor@dk5n55 ~/work/arch-pc/lab07 $
```

Как указано на следующем рисунке (рис. ??)

### 3.1 Выводы по результатам выполнения заданий :

- В этой части мы смогли узнать, как преобразовать некоторые математические идеи в реальный код на ассемблере, что помогло нам получить более глубокое представление о том, как работать с регистрами.

## 4 Выводы

- В седьмой лаборатории мы в основном научились писать программы, выполняющие арифметические операции, и научились вычислять математические выражения средней сложности.