

Отчёт по лабораторной работе №6

дисциплина: архитектура компьютера

Терещенкова Маргарита Владимировна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Символьные и численные данные в NASM	8
4.2	Выполнение арифметических операций в NASM	13
4.3	Ответы на вопросы по программе.	16
5	Выполнение самостоятельной работы	18
6	Выводы	22
	Список литературы	23

Список иллюстраций

4.1	Создание директории и файла	8
4.2	Копирование файла	8
4.3	Редактирование файла	9
4.4	Запуск исполняемого файла	9
4.5	Редактирование файла	10
4.6	Запуск исполняемого файла	10
4.7	Создание файла	10
4.8	Редактирование файла	11
4.9	Запуск исполняемого файла	11
4.10	Редактирование файла	11
4.11	Запуск исполняемого файла	12
4.12	Редактирование файла	12
4.13	Запуск исполняемого файла	12
4.14	Создание файла	13
4.15	Редактирование файла	13
4.16	Запуск исполняемого файла	14
4.17	Редактирование программы	14
4.18	Запуск исполняемого файла	14
4.19	Создание файла	15
4.20	Редактирование программы	15
4.21	Запуск исполняемого программы	16
5.1	Создание файла	18
5.2	Написание программы	19
5.3	Запуск исполняемого программы	19

1 Цель работы

Освоение 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. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`. - Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`. - Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

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 делает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

1. Создаю каталог для программ лабораторной работы № 6 (lab06) с помощью утилиты **mkdir**, перехожу в него и создаю файл lab6-1.asm с помощью утилиты **touch**.

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06/report$ mkdir ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06/report$ cd ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06/
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch lab6-1.asm
```

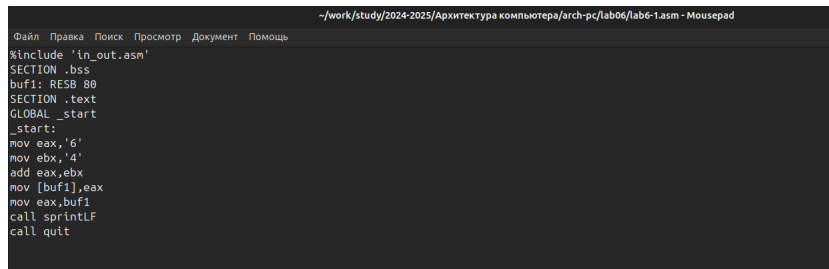
Рис. 4.1: Создание директории и файла

Копирую в текущий каталог файл in_out.asm с помощью утилиты **cp**, т.к. он будет использоваться в других программах. И проверяю наличие файла в данной директории с помощью команды **ls**.

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ cp ~/Загрузки/in_out.asm in_out.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.2: Копирование файла

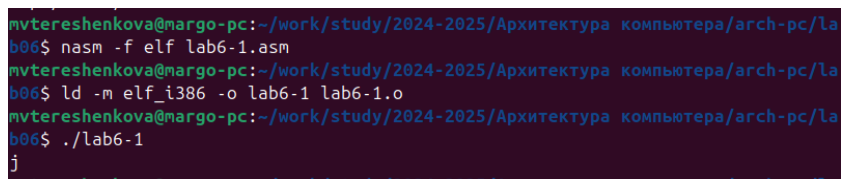
2. Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра **eax**.



```
~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06/lab6-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
%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. 4.3: Редактирование файла

Создаю исполняемый файл программы и запускаю его.



```
mvtreshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-1.asm
mvtreshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
mvtreshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-1
j
```

Рис. 4.4: Запуск исполняемого файла

Вывод программы: символ **j**, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

- 3. Изменяю текст программы и вместо символов, запишем в регистры числа (“6” и “4” на цифры 6 и 4).

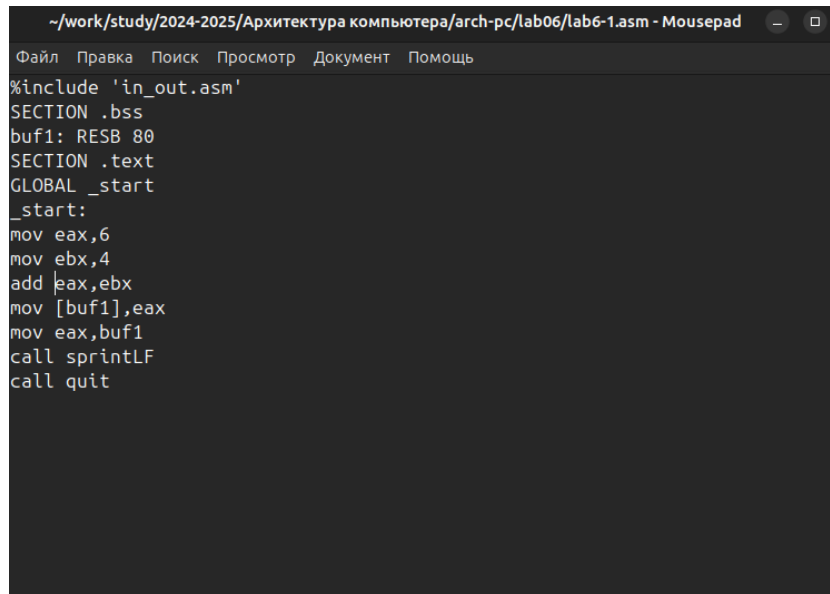


Рис. 4.5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его.

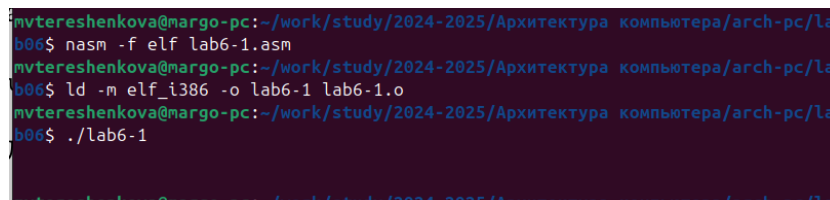


Рис. 4.6: Запуск исполняемого файла

Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

4. Создаю новый файл lab6-2.asm с помощью утилиты **touch**.

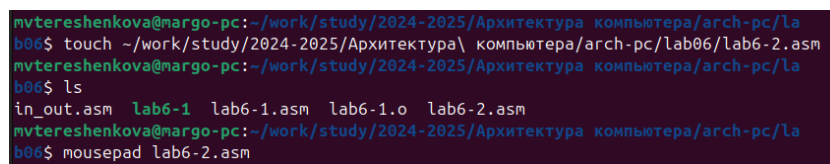
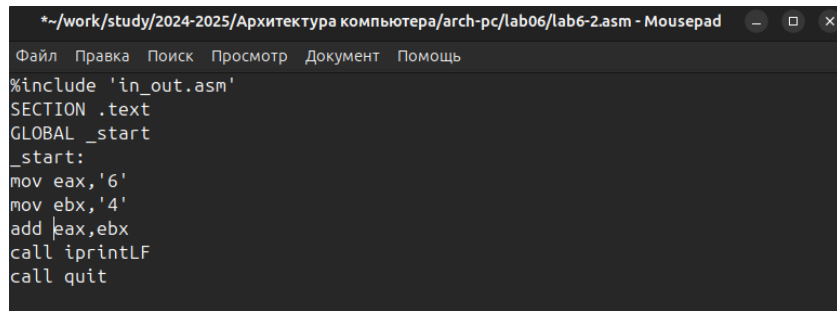


Рис. 4.7: Создание файла

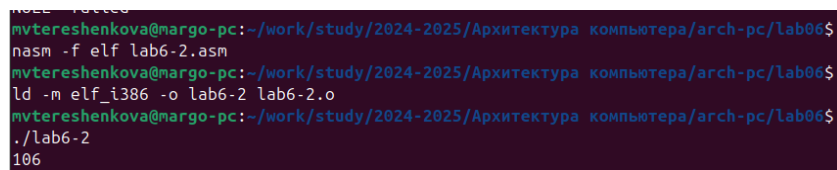
Ввожу в файл текст другой программы для вывода значения регистра eax.

A screenshot of a text editor window titled '*~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06/lab6-2.asm - Mousepad'. The editor contains the following assembly code:

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 4.8: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2.

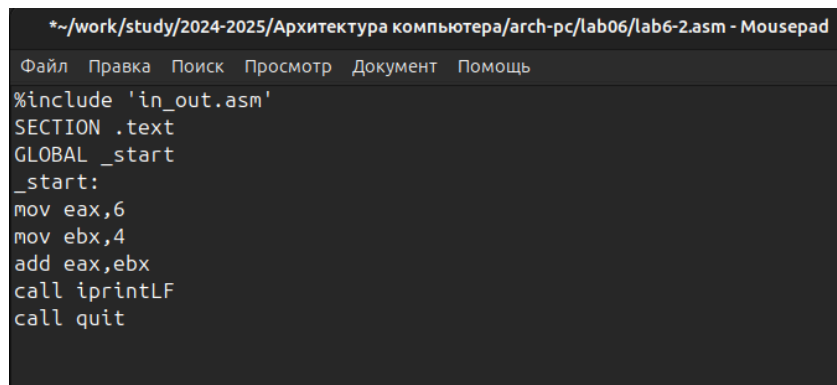
A screenshot of a terminal window showing the following commands and output:

```
mvtreshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
nasm -f elf lab6-2.asm
mvtreshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
ld -m elf_i386 -o lab6-2 lab6-2.o
mvtreshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
./lab6-2
106
```

Рис. 4.9: Запуск исполняемого файла

Теперь выводится число **106**, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

5. Аналогично предыдущему примеру изменила символы на числа. (“6” и “4” на 6 и 4).

A screenshot of a text editor window titled '*~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06/lab6-2.asm - Mousepad'. The editor contains the following modified assembly code:

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 4.10: Редактирование файла

Создаю и запускаю новый исполняемый файл.

```

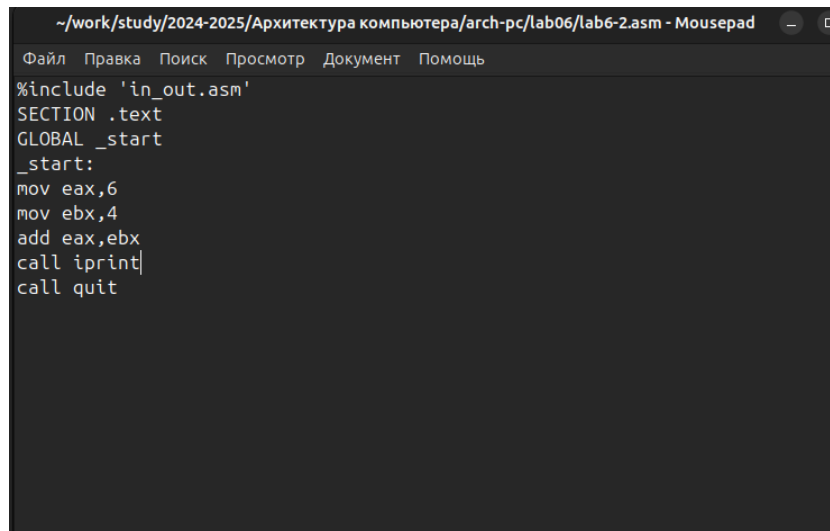
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
nasm -f elf lab6-2.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
ld -m elf_i386 -o lab6-2 lab6-2.o
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
./lab6-2
10

```

Рис. 4.11: Запуск исполняемого файла

Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

Заменяю в тексте программы функцию **iprintLF** на **iprint**.



```

~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06/lab6-2.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit

```

Рис. 4.12: Редактирование файла

Создаю и запускаю новый исполняемый файл.

```

mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
nasm -f elf lab6-2.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
ld -m elf_i386 -o lab6-2 lab6-2.o
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
./lab6-2
10
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$

```

Рис. 4.13: Запуск исполняемого файла

iprint не добавляет к выводу символ переноса строки, в отличие от **iprintLF**.

4.2 Выполнение арифметических операций в NASM

6. Создаю файл lab6-3.asm с помощью утилиты **touch**.

```
ютера/arch-pc/lab06$ touch lab6-3.asm

mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютер
epa/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6
-2.o lab6-3.asm
```

Рис. 4.14: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$

```
*~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06/lab6-3.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi
call iprintLF ; из 'edi' в виде символов
mov eax,rem
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.15: Редактирование файла

Создаю исполняемый файл и запускаю его.

```

mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютер
ера/arch-pc/lab06$ nasm -f elf lab6-3.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютер
ера/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютер
ера/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1

```

Рис. 4.16: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$.

```

*~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06/lab6-3.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
mov eax,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi
call iprintLF ; из 'edi' в виде символов
mov eax,rem
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 4.17: Редактирование программы

Создаю и запускаю новый исполняемый файл.

```

mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютер
ера/arch-pc/lab06$ nasm -f elf lab6-3.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютер
ера/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютер
ера/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1

```

Рис. 4.18: Запуск исполняемого файла

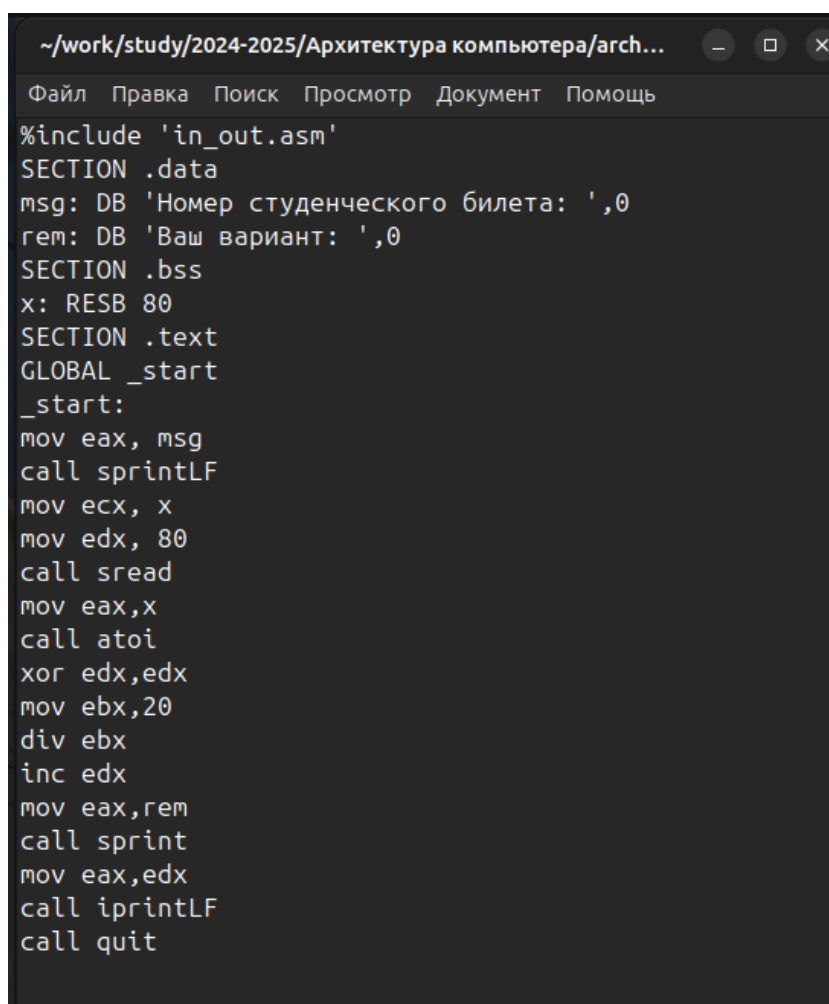
Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

7. Создаю файл `numbervariant.asm` с помощью утилиты **touch**.

```
mvertereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютер  
epa/arch-pc/lab06$ touch numbervariant.asm  
mvertereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютер  
epa/arch-pc/lab06$ mousepad numbervariant.asm
```

Рис. 4.19: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета.



```
~/work/study/2024-2025/Архитектура компьютера/arch...  
Файл Правка Поиск Просмотр Документ Помощь  
%include 'in_out.asm'  
SECTION .data  
msg: DB 'Номер студенческого билета: ',0  
rem: DB 'Ваш вариант: ',0  
SECTION .bss  
x: RESB 80  
SECTION .text  
GLOBAL _start  
_start:  
mov eax, msg  
call sprintf  
mov ecx, x  
mov edx, 80  
call sread  
mov eax, x  
call atoi  
xor edx, edx  
mov ebx, 20  
div ebx  
inc edx  
mov eax, rem  
call sprintf  
mov eax, edx  
call iprintLF  
call quit
```

Рис. 4.20: Редактирование программы

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры.

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch
-pc/lab06$ nasm -f elf numbervariant.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch
-pc/lab06$ ld -m elf_i386 -o numbervariant numbervariant.o
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch
-pc/lab06$ ./numbervariant
Номер студенческого билета:
1132246723
Ваш вариант: 4
```

Рис. 4.21: Запуск исполняемого программы

Программа вывела, что мой вариант - 4.

4.3 Ответы на вопросы по программе.

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

mov eax,rem

call sprint

2. **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 самостоятельной работы

Создаю файл lab6-4.asm .

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch
-pc/lab06$ touch lab6-4.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch
-pc/lab06$ ls
in_out.asm  lab6-1.o    lab6-2.o    lab6-3.o    numbervariant.asm
lab6-1      lab6-2      lab6-3      lab6-4.asm  numbervariant.o
lab6-1.asm  lab6-2.asm  lab6-3.asm  numbervariant
```

Рис. 5.1: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $\frac{4}{3} \cdot (x-1) + 5$. Это выражение было под вариантом 4.

```

*~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06/lab6-4.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
%include 'in_out.asm'
SECTION .data
    msg db 'Введите значение переменной x: ', 0
    rem db 'Результат: ', 0

SECTION .bss
    x RESB 80

SECTION .text
    GLOBAL _start
_start:
    mov eax, msg
    call sprint

    mov ecx, x
    mov edx, 80
    call sread

    mov eax, x
    call atoi

    ; f(x) = (3 / 4) * (x - 1) + 5
    sub eax, 1    ; eax = x - 1
    mov ebx, 3    ; Умножаем на 3
    imul eax, ebx ; eax = (x - 1) * 3
    mov ebx, 4    ; Делим на 4
    xor edx, edx  ; Обнуляем edx перед делением
    div ebx       ; eax = (x - 1) * 3 / 4
    add eax, 5    ; eax = (x - 1) * 3 / 4 + 5

    mov edi, eax
    mov eax, rem
    call sprint
    mov eax, edi
    call iprintLF
    call quit

```

Рис. 5.2: Написание программы

Создаю и запускаю исполняемый файл 2 раза. Ввожу значение переменных x , которые написаны в моём варианте, а именно $x1=4$ и $x2=10$.

```

mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch
-pc/lab06$ nasm -f elf lab6-4.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch
-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch
-pc/lab06$ ./lab6-4
Введите значение переменной x: 4
Результат: 7
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch
-pc/lab06$ ./lab6-4
Введите значение переменной x: 10
Результат: 11

```

Рис. 5.3: Запуск исполняемого программы

Программа выводит верный ответ, учитывая, что при выполнении деления в качестве результата можно использовать только целую часть от деления и не учитывать остаток (т.е. $5 : 2 = 2$).

Листинг Программа для вычисления значения выражения $4/3*(x-1)+5$.

```
%include 'in_out.asm'

SECTION .data
msg db 'Введите значение переменной x:', 0
rem db 'Результат:', 0

SECTION .bss
x RESB 80

SECTION .text
GLOBAL _start

_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
;  $f(x) = (3 / 4) * (x - 1) + 5$ 
sub eax, 1 ;  $eax = x - 1$ 
mov ebx, 3 ; Умножаем на 3
imul eax, ebx ;  $eax = (x - 1) * 3$ 
mov ebx, 4 ; Делим на 4
xor edx, edx ; Обнуляем edx перед делением
div ebx ;  $eax = (x - 1) * 3 / 4$ 
add eax, 5 ;  $eax = (x - 1) * 3 / 4 + 5$ 
mov edi, eax
```

```
mov eax, rem  
call sprint  
mov eax, edi  
call iprintLF  
call quit
```

6 Выводы

Благодаря данной лабораторной работе освоила арифметических инструкций языка ассемблера NASM.

Список литературы

1. Лабораторная работа №6
2. Таблица ASCII