

Отчёт по лабораторной работе №6

дисциплина: Архитектура компьютера

Баранова Анна Андреевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	9
4.1	Символьные и численные данные в NASM	9
4.2	Выполнение арифметических операций в NASM	15
4.3	Задание для самостоятельной работы	21
5	Выводы	23

List of Figures

4.1	Создание каталога для программ лабораторной работы № 6, переход в него и создание файла lab6-1.asm	9
4.2	Изменения в файле lab6-1.asm	9
4.3	Копирование файла in_out.asm в каталог с файлом lab6-1.asm с помощью функциональной клавиши F5	10
4.4	Копирование файла in_out.asm в каталог с файлом lab6-1.asm с помощью функциональной клавиши F5	10
4.5	Создание исполняемого файла и его запуск	11
4.6	Создание исполняемого файла и его запуск	11
4.7	Изменения в файле lab6-1.asm	11
4.8	Создание исполняемого файла и его запуск	12
4.9	Создание файла lab6-2.asm	12
4.10	Создание файла lab6-2.asm	12
4.11	Изменения в файле lab6-2.asm	13
4.12	Создание исполняемого файла и его запуск	13
4.13	Создание исполняемого файла и его запуск	13
4.14	Изменения в файле lab6-2.asm	14
4.15	Создание исполняемого файла и его запуск	14
4.16	Изменения в файле lab6-2.asm	15
4.17	Создание исполняемого файла и его запуск	15
4.18	Создание файла lab6-3.asm	15
4.19	Создание файла lab6-3.asm	16
4.20	Изменения в файле lab6-3.asm	16
4.21	Создание исполняемого файла и его запуск	17
4.22	Создание исполняемого файла и его запуск	17
4.23	Изменения в файле lab6-3.asm	17
4.24	Создание исполняемого файла и его запуск	18
4.25	Создание файла variant.asm	18
4.26	Создание файла variant.asm	18
4.27	Изменения в файле variant.asm	19
4.28	Создание исполняемого файла и его запуск	19
4.29	Создание исполняемого файла и его запуск	19
4.30	Написание программы вычисления выражения $\square = \square(\square$	21
4.31	Написание программы вычисления выражения $\square = \square(\square$	21
4.32	Создание исполняемого файла и проверка его работы	22
4.33	Создание исполняемого файла и проверка его работы	22

1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

2 Задание

В ходе выполнения данной лабораторной работы необходимо изучить:

- Символьные и численные данные в NASM;
- как выполнять арифметические операции в NASM;
- адресация в NASM.

Выполнив эту работу, мы освоим арифметические инструкции языка ассемблера NASM.

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. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации.

Существует три основных способа адресации:

- 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.

Схема команды целочисленного сложения `add` (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда `add` работает как с числами со знаком, так и без знака.

Команда целочисленного вычитания `sub` (от англ. subtraction – вычитание) работает аналогично команде `add`.

Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание – декрементом. Для этих операций существуют специальные коман-

ды: `inc` (от англ. `increment`) и `dec` (от англ. `decrement`), которые увеличивают и уменьшают на 1 свой операнд.

Еще одна команда, которую можно отнести к арифметическим командам это команда изменения знака `neg`. Команда `neg` рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.

Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды. Для беззнакового умножения используется команда `mul` (от англ. `multiply` – умножение). Для знакового умножения используется команда `imul`.

Для деления, как и для умножения, существует 2 команды `div` (от англ. `divide` – деление) и `idiv`.

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.

Расширенная таблица ASCII состоит из двух частей. Первая (символы с кодами 0-127) является универсальной (см. Приложение.), а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться.

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.

Для выполнения лабораторных работ в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно.

4 Выполнение лабораторной работы

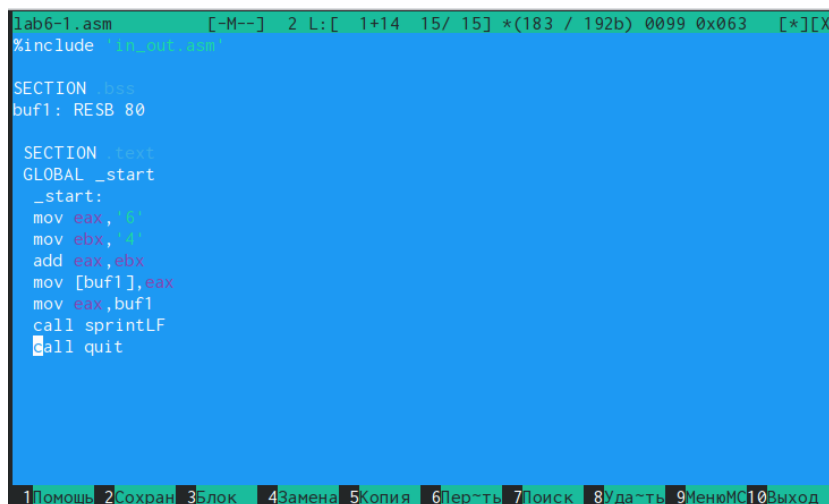
4.1 Символьные и численные данные в NASM

Создадим каталог для программ лабораторной работы № 6, перейдём в него и создадим файл lab6-1.asm (рис. 4.1).

```
aabaranova@dk3n55 ~ $ mkdir ~/work/arch-pc/lab06
aabaranova@dk3n55 ~ $ cd ~/work/arch-pc/lab06
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ touch lab6-1.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 4.1: Создание каталога для программ лабораторной работы № 6, переход в него и создание файла lab6-1.asm

Введём в файл lab6-1.asm текст программы (рис. 4.2).



```
lab6-1.asm  [-M--]  2  L: [ 1+14 15/ 15] *(183 / 192b) 0099 0x063  [*][X]
#include "io.h"

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. 4.2: Изменения в файле lab6-1.asm

Перед созданием исполняемого файла создадим копию файла in_out.asm в каталоге ~/work/arch-pc/lab06. (рис. 4.3), (рис. 4.4).

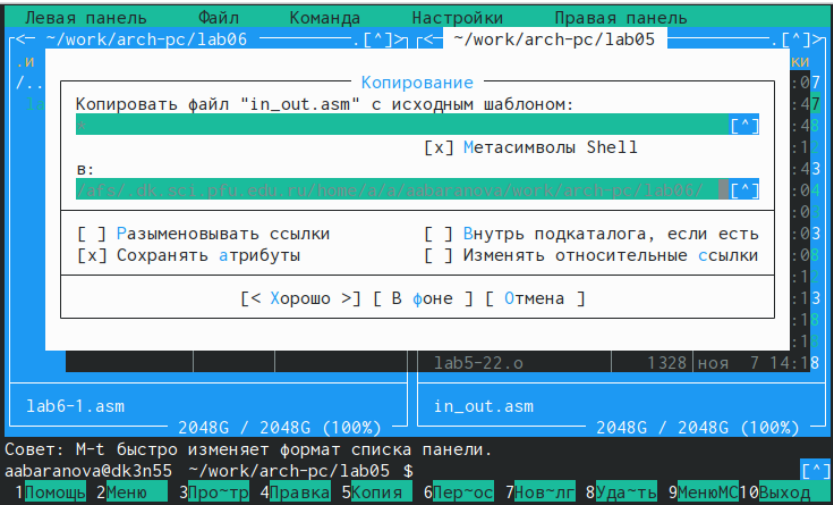


Рис. 4.3: Копирование файла in_out.asm в каталог с файлом lab6-1.asm с помощью функциональной клавиши F5

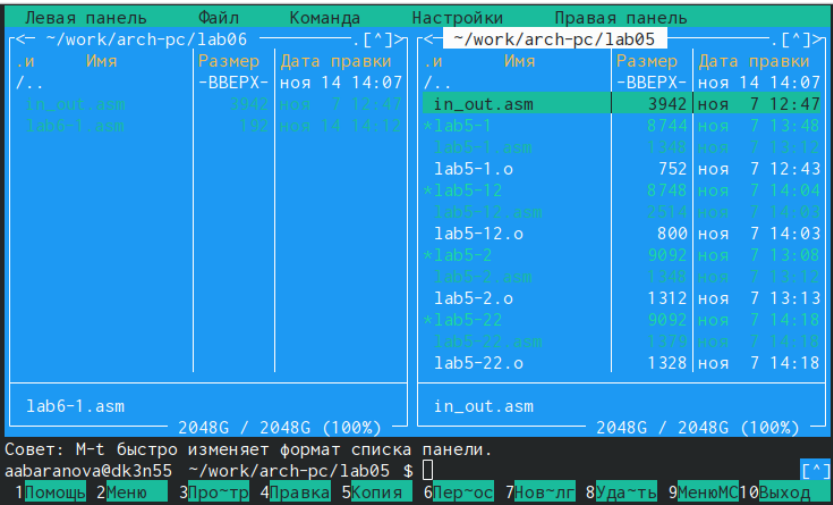


Рис. 4.4: Копирование файла in_out.asm в каталог с файлом lab6-1.asm с помощью функциональной клавиши F5

Создадим исполняемый файл и запустим его (рис. 4.5), (рис. 4.6).

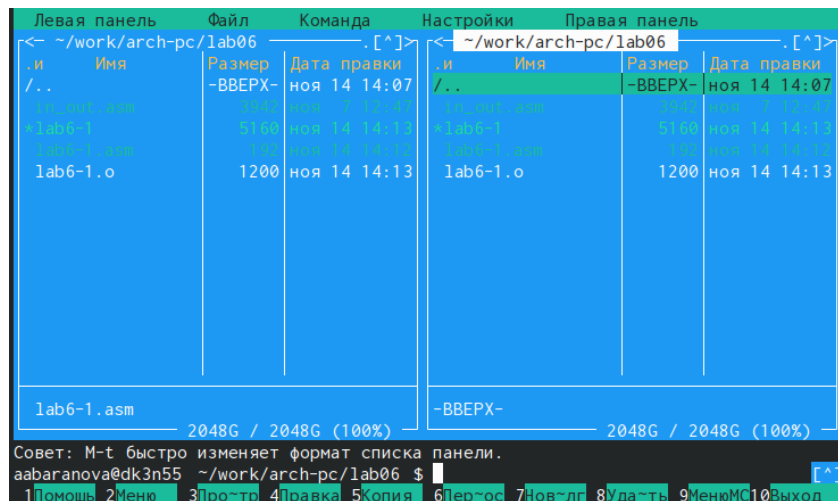


Рис. 4.5: Создание исполняемого файла и его запуск

```
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ mc
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-1
j
aabaranova@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 4.6: Создание исполняемого файла и его запуск

Изменим текст программы и вместо символов, запишем в регистры числа (рис. 4.7).

```
lab6-1.asm [-M--] 11 L: [ 1+14 15/ 15] *(188 / 188b) <EOF> [*][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
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМс 10Выход

Рис. 4.7: Изменения в файле lab6-1.asm

Создадим исполняемый файл и запустим его (рис. 4.8).

```
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-1
aabaranova@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 4.8: Создание исполняемого файла и его запуск

Создадим файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введём в него текст программы (рис. 4.9), (рис. 4.10), (рис. 4.11).

```
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 4.9: Создание файла lab6-2.asm

Левая панель				Файл	Команда	Настройки				Правая панель			
< ~/work/arch-pc/lab06				.	[^]>	< ~/work/arch-pc/lab06				.	[^]>		
.и		Имя		Размер	Дата правки	.и		Имя		Размер	Дата правки		
/..				-ВВЕРХ-	ноя 14 14:07	/..				-ВВЕРХ-	ноя 14 14:07		
lab_out.asm				3042	ноя 7 12:47	lab_out.asm				3042	ноя 7 12:47		
lab6-1				5168	ноя 14 14:24	lab6-1				5168	ноя 14 14:24		
lab6-1.asm				188	ноя 14 14:23	lab6-1.asm				188	ноя 14 14:23		
lab6-1.o				1200	ноя 14 14:23	lab6-1.o				1200	ноя 14 14:23		
lab6-2.asm				0	ноя 14 14:27	lab6-2.asm				0	ноя 14 14:27		
lab6-1.asm								-ВВЕРХ-					
2048G / 2048G (100%)								2048G / 2048G (100%)					
Совет: M-t быстро изменяет формат списка панели.													
aabaranova@dk3n55 ~/work/arch-pc/lab06 \$													
1Помощь		2Меню		3Протр		4Правка		5Копия		6Пернос			
7Новлг		8Уда-ть		9МенюМС		10Выход							

Рис. 4.10: Создание файла lab6-2.asm

```

lab6-2.asm [-M--] 2 L: [ 1+11 12/ 12] *(125 / 134b) 0099 0x063 [*][X]
#include "in_out.asm"

SECTION .text
GLOBAL _start
_start:

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

call quit

```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход

Рис. 4.11: Изменения в файле lab6-2.asm

Создадим исполняемый файл и запустим его (рис. 4.12), (рис. 4.13).



Рис. 4.12: Создание исполняемого файла и его запуск

```

aabaranova@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm

aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o

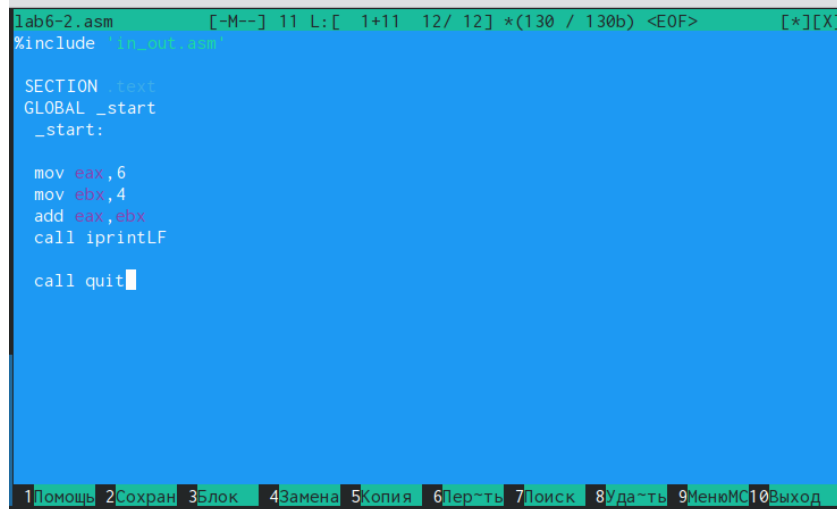
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-2
106

aabaranova@dk3n55 ~/work/arch-pc/lab06 $

```

Рис. 4.13: Создание исполняемого файла и его запуск

Изменим текст программы и вместо символов, запишем в регистры числа (рис. 4.14).



```
lab6-2.asm [-M--] 11 L: [ 1+11 12/ 12] *(130 / 130b) <EOF> [*][X]
#include "in_out.asm"

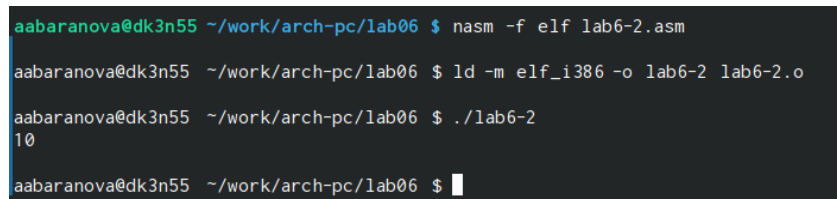
SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 4.14: Изменения в файле lab6-2.asm

Создадим исполняемый файл и запустим его (рис. 4.15).



```
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-2
10
aabaranova@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 4.15: Создание исполняемого файла и его запуск

Заменяем функцию iprintLF на iprint. Создадим исполняемый файл и запустим его (рис. 4.16), (рис. 4.17).

```
lab6-2.asm [-M--] 13 L: [ 1+ 9 10/ 12] *(115 / 128b) 0010 0x00A [*][X]
#include "in_out.asm"

SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 4.16: Изменения в файле lab6-2.asm

```
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-2
10
aabaranova@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 4.17: Создание исполняемого файла и его запуск

4.2 Выполнение арифметических операций в NASM

Создадим файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 (рис. 4.18), (рис. 4.19).

```
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 4.18: Создание файла lab6-3.asm



Рис. 4.19: Создание файла lab6-3.asm

Введём в файл lab6-3.asm текст программы (рис. 4.20).

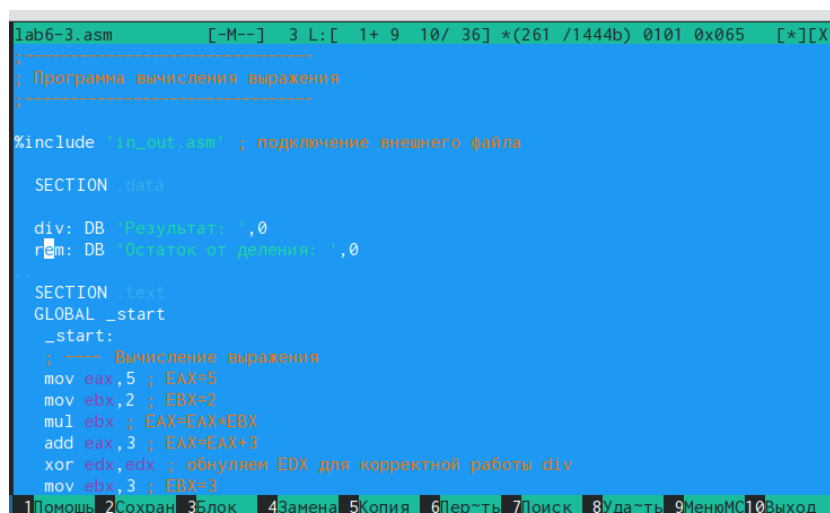


Рис. 4.20: Изменения в файле lab6-3.asm

Создадим исполняемый файл и запустим его (рис. 4.21), (рис. 4.22).



Рис. 4.21: Создание исполняемого файла и его запуск

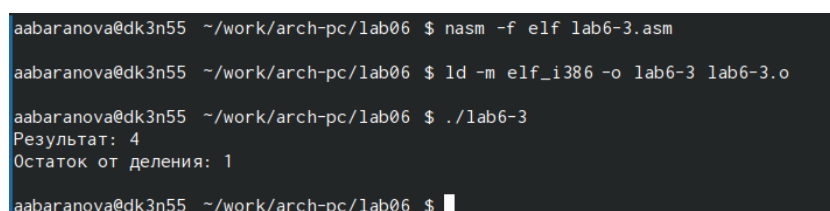


Рис. 4.22: Создание исполняемого файла и его запуск

Изменим текст программы для вычисления выражения $\square(\square) = (4 \square 6 + 2)/5$ (рис. 4.23).

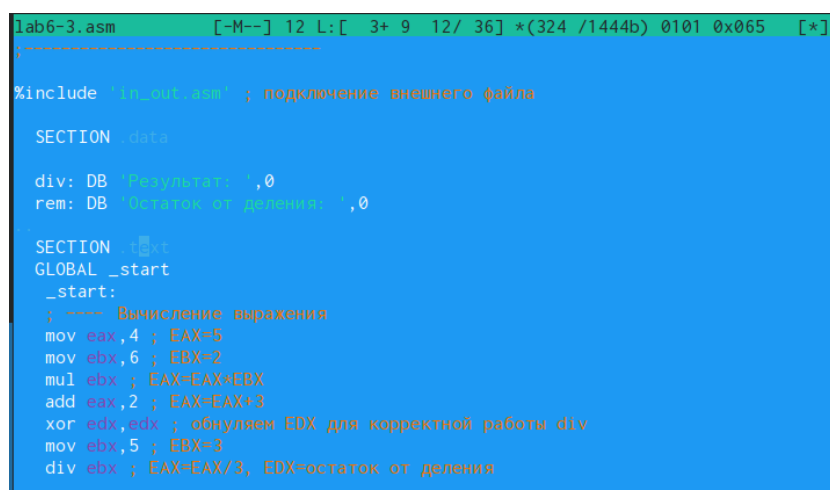


Рис. 4.23: Изменения в файле lab6-3.asm

Создадим исполняемый файл и запустим его (рис. 4.24).

```
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
aabaranova@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 4.24: Создание исполняемого файла и его запуск

Создадим файл variant.asm в каталоге ~/work/arch-pc/lab06 (рис. 4.25), (рис. 4.26).

```
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 4.25: Создание файла variant.asm



Рис. 4.26: Создание файла variant.asm

Введём в файл lab6-3.asm текст программы (рис. 4.27).

```

variant.asm      [-M--] 1 L: [ 1+ 0 1/ 37] *(1 / 692b) 0045 0x02D  [*] [X]
;
; Программа вычисления варианта
;
%include "in_out.asm"

SECTION .data
msg: DB "Введите № студенческого билета: ",0
rem: DB "Ваш вариант: ",0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintfLF

```

Рис. 4.27: Изменения в файле variant.asm

Создадим исполняемый файл и запустим его (рис. 4.28), (рис. 4.29).

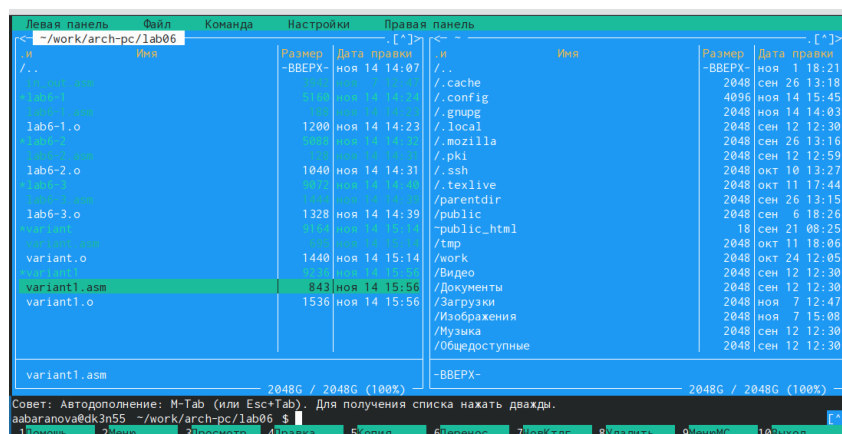


Рис. 4.28: Создание исполняемого файла и его запуск

```

aabaranova@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132246811
Ваш вариант: 12
aabaranova@dk3n55 ~/work/arch-pc/lab06 $

```

Рис. 4.29: Создание исполняемого файла и его запуск

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
```



```

aabaranova@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf variant1.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant1 variant1.o
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ./variant1
Выражение для вычисления: y = (8x-6)/2
Введите значение переменной x:
1
Результат:
1
aabaranova@dk3n55 ~/work/arch-pc/lab06 $

```

Рис. 4.32: Создание исполняемого файла и проверка его работы

```

aabaranova@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf variant1.asm
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant1 variant1.o
aabaranova@dk3n55 ~/work/arch-pc/lab06 $ ./variant1
Выражение для вычисления: y = (8x-6)/2
Введите значение переменной x:
5
Результат:
17
aabaranova@dk3n55 ~/work/arch-pc/lab06 $

```

Рис. 4.33: Создание исполняемого файла и проверка его работы

5 Выводы

В ходе выполнения данной лабораторной работы были освоены арифметические инструкции языка ассемблера NASM.