

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: Архитектура компьютера

Студент: Шабакова Карина Баировна

Группа: НКАбд-05-24

МОСКВА

2024 г.

1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

2 Задание

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

3 Теоретическое введение

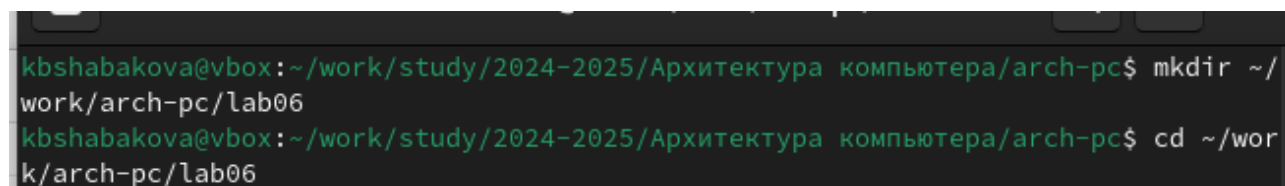
Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена 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 сделает 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

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. 1). Перехожу в созданный каталог с помощью утилиты `cd`.



```
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ mkdir ~/work/arch-pc/lab06
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd ~/work/arch-pc/lab06
```

Рис. 1: Создание директории

С помощью утилиты `touch` создаю файл `lab6-1.asm` (рис. 2).

```

kbshabakova@vbox:~/work/arch-pc/lab06$ touch lab6-1.asm
kbshabakova@vbox:~/work/arch-pc/lab06$ ды
bash: ды: команда не найдена...
kbshabakova@vbox:~/work/arch-pc/lab06$ ls
lab6-1.asm

```

Рис. 2: Создание файла

Копирую в текущий каталог файл in_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. 3).

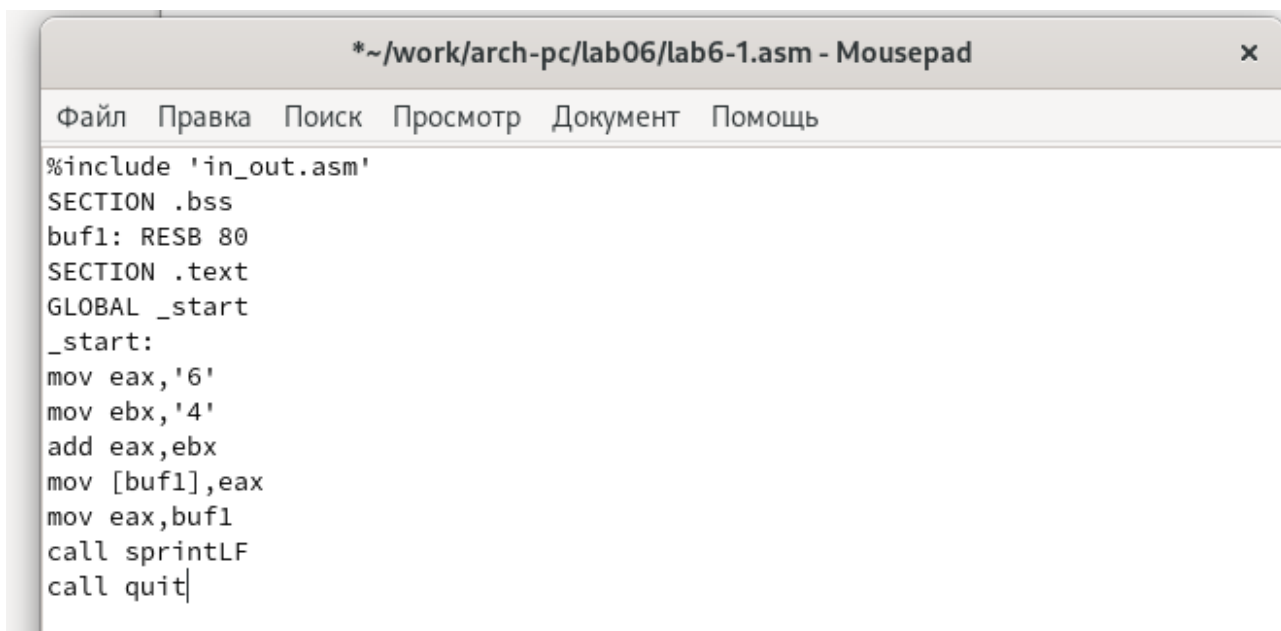
```

kbshabakova@vbox:~/work/arch-pc/lab06$ cp ~/Загрузки/in_out.asm in_out.asm
kbshabakova@vbox:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm

```

Рис. 3: Создание копии файла

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. 4).



```

*~/work/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 sprintLF
call quit

```

Рис. 4: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 5). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII

```

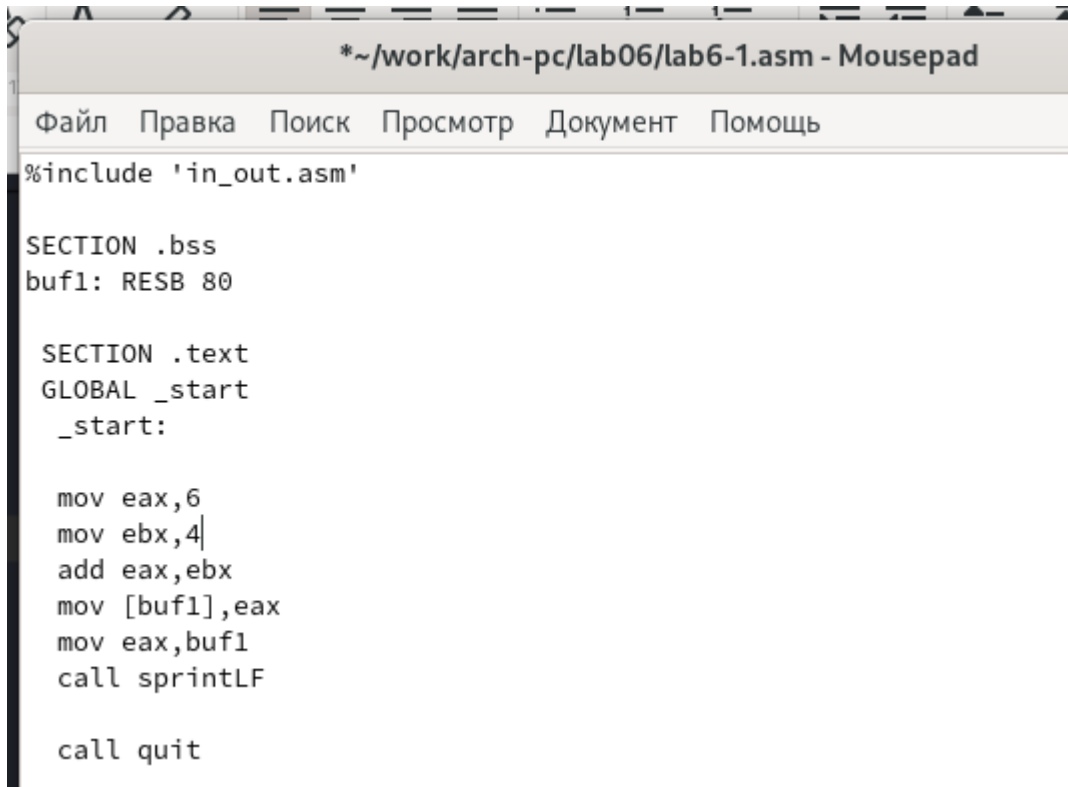
kbshabakova@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
kbshabakova@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
kbshabakova@vbox:~/work/arch-pc/lab06$ ./lab6-1
j
kbshabakova@vbox:~/work/arch-pc/lab06$

```

сумме двоичных кодов символов 4 и 6.

Рис. 5: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 6).



```
*~/work/arch-pc/lab06/lab6-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include 'in_out.asm'

SECTION .bss
buf1: RESB 80

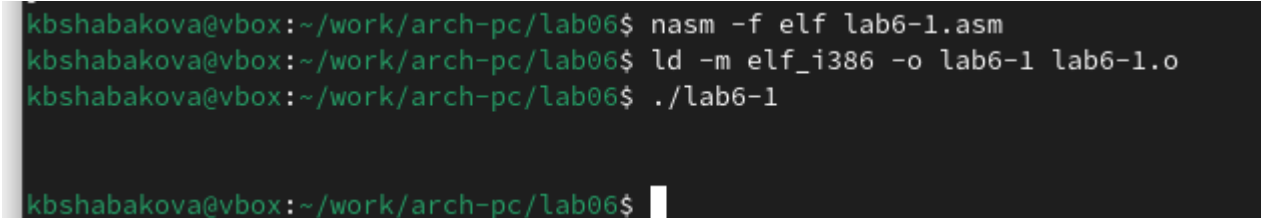
SECTION .text
GLOBAL _start
_start:

    mov eax,6
    mov ebx,4
    add eax,ebx
    mov [buf1],eax
    mov eax,buf1
    call sprintLF

    call quit
```

Рис. 6: Редактирование файла

Создаю новый исполняемый файл

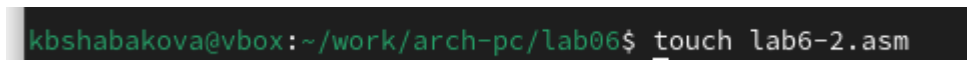


```
kbshabakova@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
kbshabakova@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
kbshabakova@vbox:~/work/arch-pc/lab06$ ./lab6-1
kbshabakova@vbox:~/work/arch-pc/lab06$
```

Символ не отображается при выводе на экран.

Рис. 7: Запуск исполняемого файла

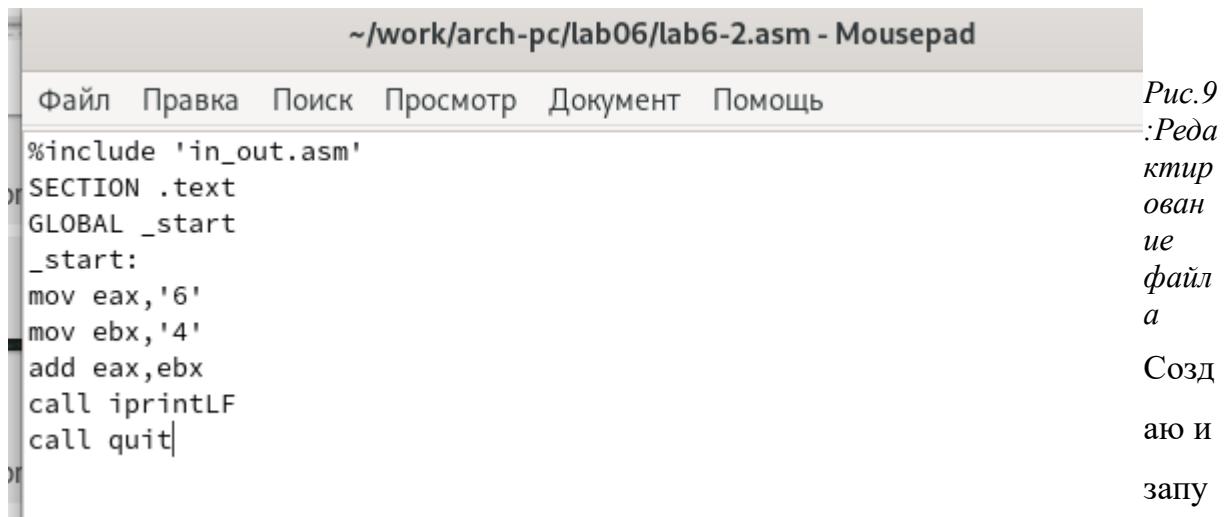
Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 8).



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

Рис. 8: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 9).



скаю исполняемый файл lab6-2 (рис. 10). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

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

Рис. 10: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 11).

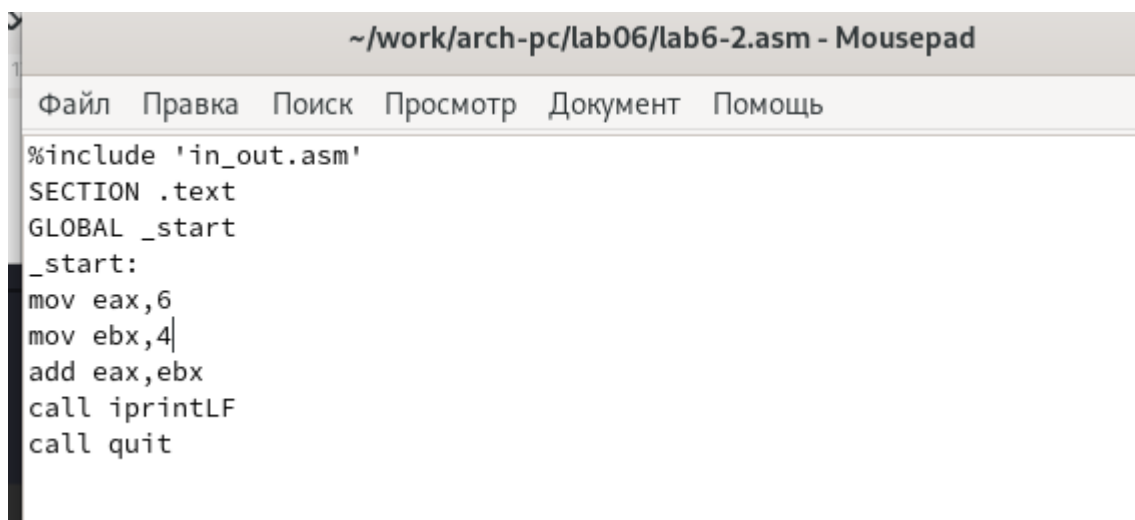


Рис.11:Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 12).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```

100
kbshabakova@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
kbshabakova@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
kbshabakova@vbox:~/work/arch-pc/lab06$ ./lab6-2
10
kbshabakova@vbox:~/work/arch-pc/lab06$

```

Рис. 12: Запуск исполняемого файла

Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. 13).

```

GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit

```

Рис. 13: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 14). Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией `iprintLF`, а `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF`.

```

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

```

Рис. 14: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью утилиты `touch` (рис. 15).

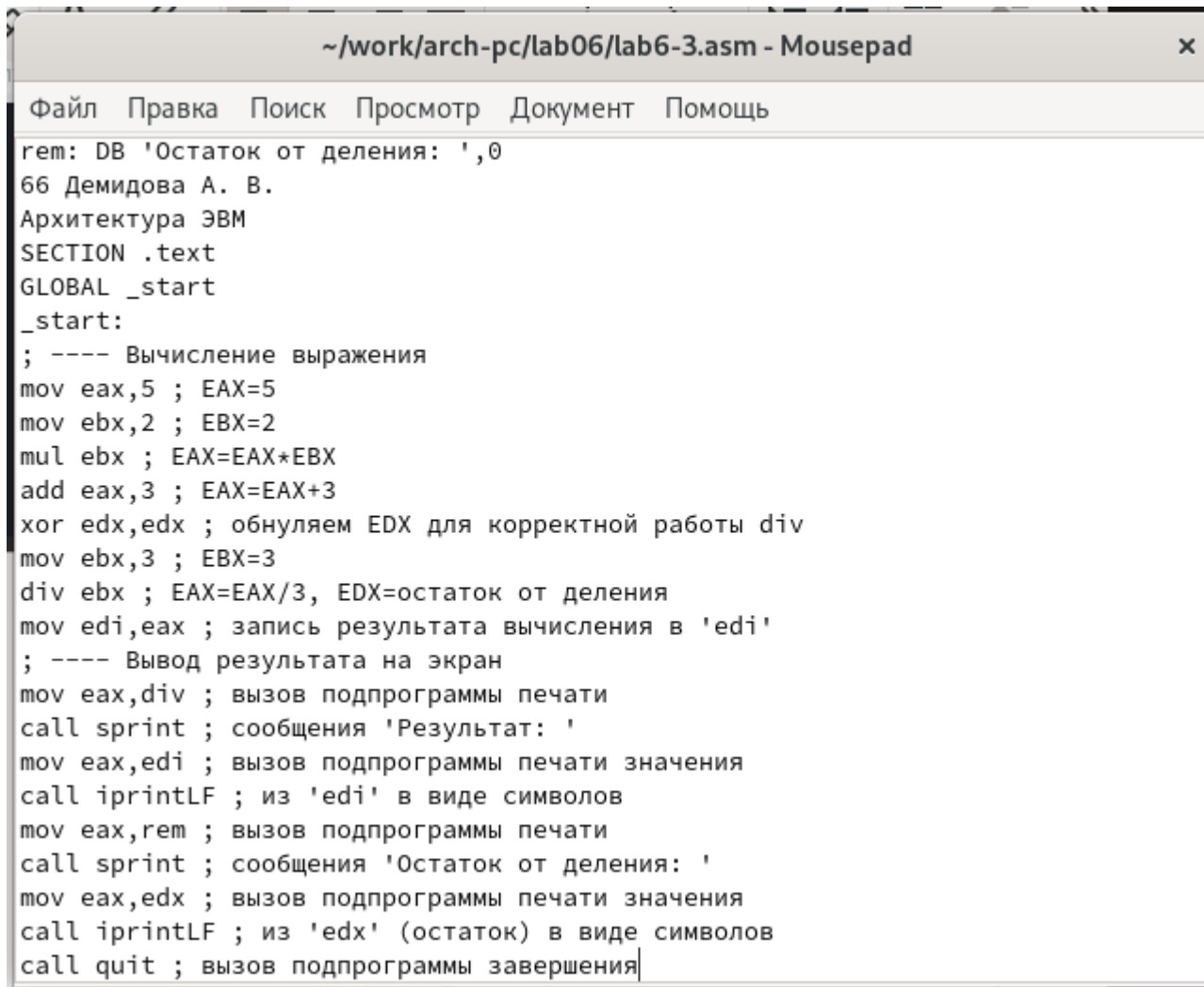
```

kbshabakova@vbox:~/work/arch-pc/lab06$ ./lab6-2
kbshabakova@vbox:~/work/arch-pc/lab06$ touch lab6-3.asm
kbshabakova@vbox:~/work/arch-pc/lab06$

```

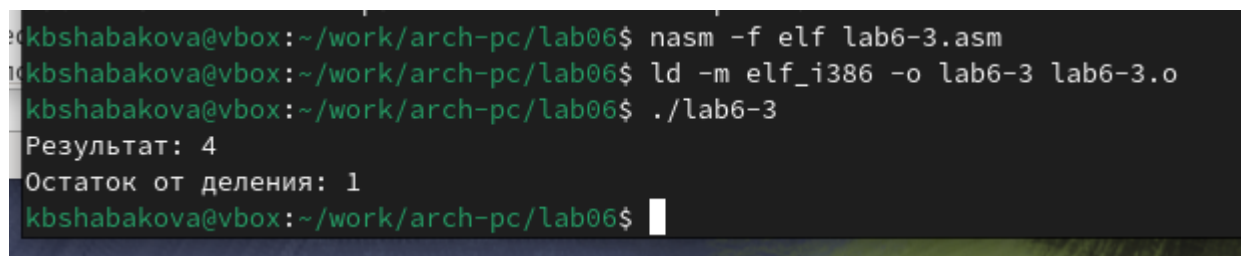
Рис. 15: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 16).



```
rem: DB 'Остаток от деления: ',0
66 Демидова А. В.
Архитектура ЭВМ
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
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'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
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. 16: Редактирование файла

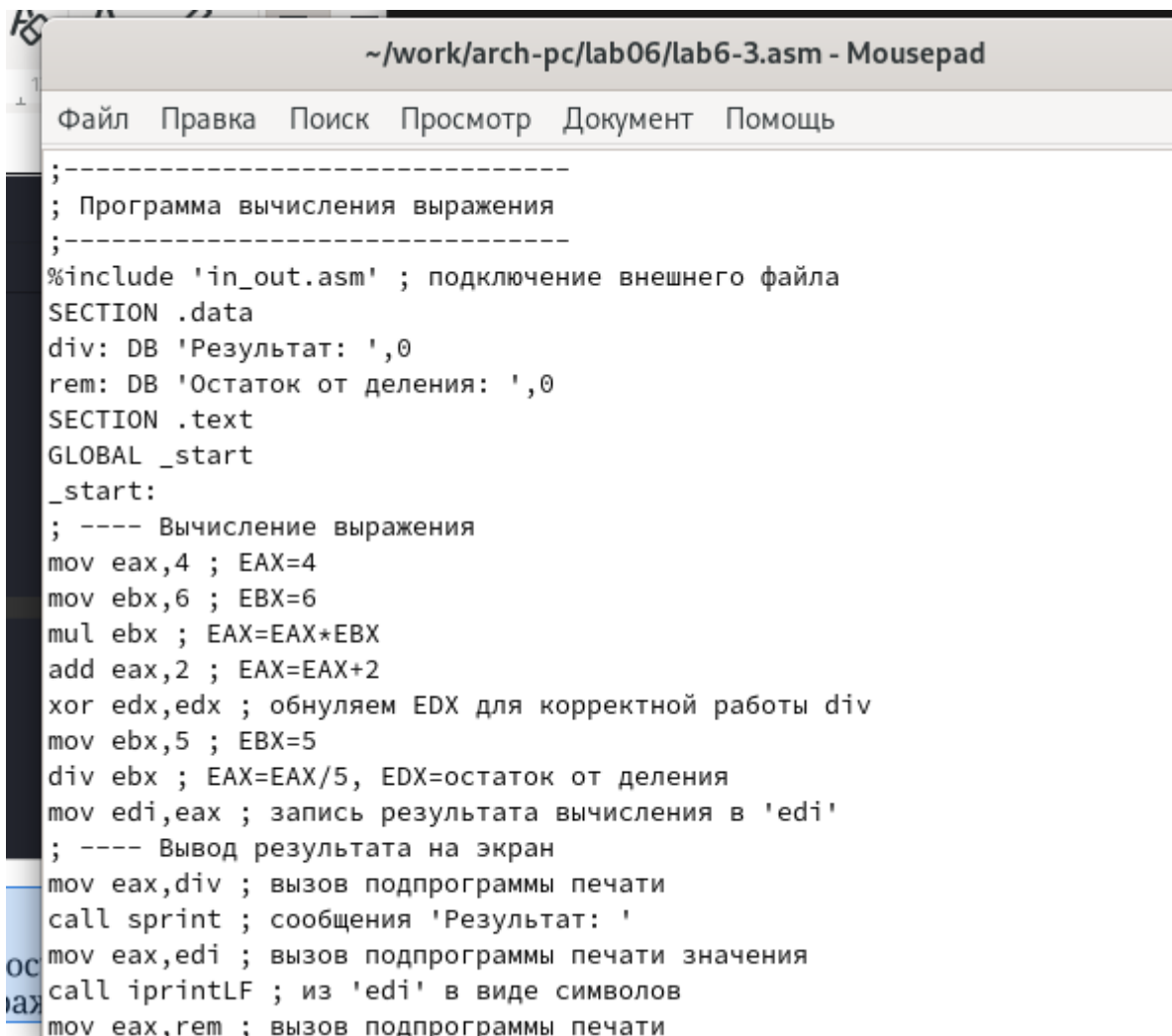


```
kbshabakova@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
kbshabakova@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
kbshabakova@vbox:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
kbshabakova@vbox:~/work/arch-pc/lab06$
```

Создаю исполняемый файл и запускаю его (рис. 17).

Рис. 17: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 18).

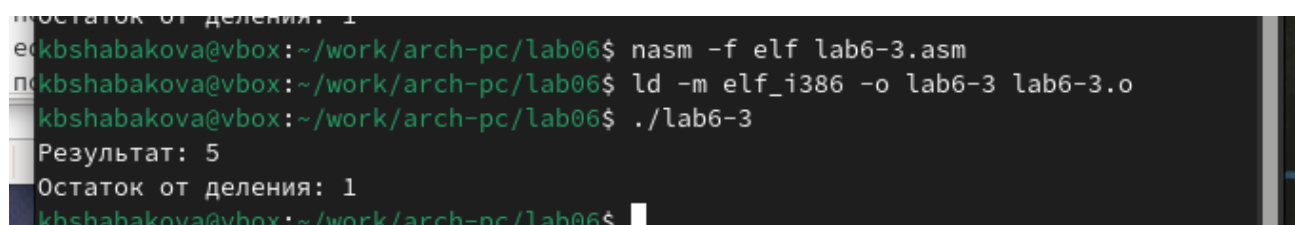


```
~/work/arch-pc/lab06/lab6-3.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

;-----
; Программа вычисления выражения
;-----
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
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'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
```

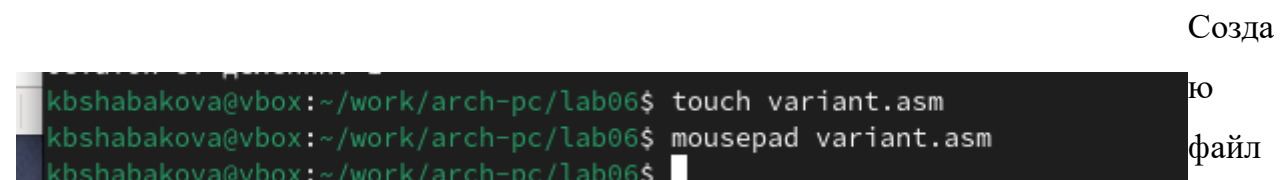
Рис. 18: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 19). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.



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

Рис. 19: Запуск исполняемого файла

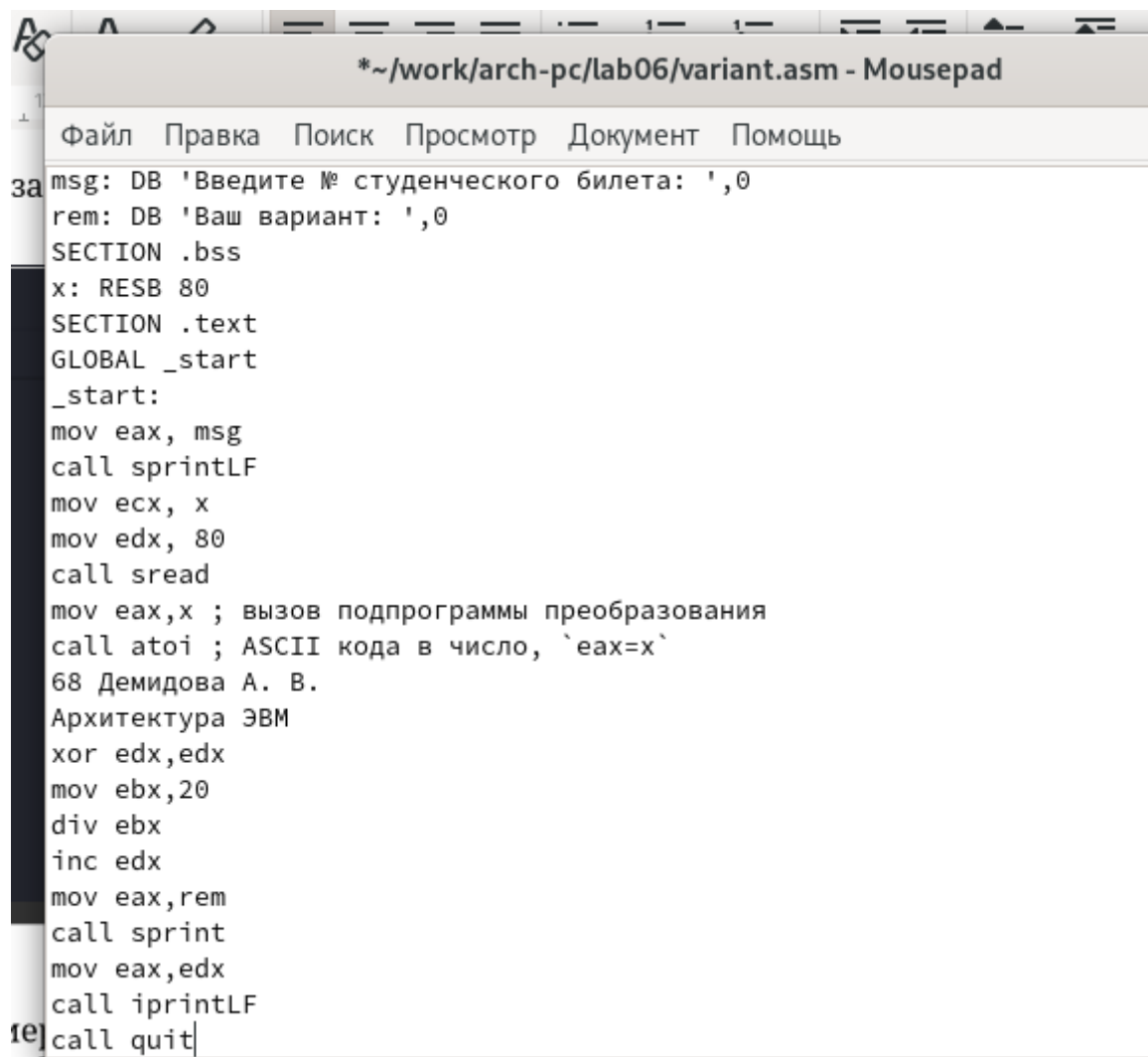


```
kbshabakova@vbox:~/work/arch-pc/lab06$ touch variant.asm
kbshabakova@vbox:~/work/arch-pc/lab06$ mousepad variant.asm
kbshabakova@vbox:~/work/arch-pc/lab06$
```

variant.asm с помощью утилиты touch (рис. 20).

Рис. 20: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру



```
*~/work/arch-pc/lab06/variant.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
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 sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
68 Демидова А. В.
Архитектура ЭВМ
xor edx,edx
mov ebx,20
div ebx
inc edx
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

студ
енче
ског
о
бил
ета
(рис
.
21).

Рис. 21: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 22). Ввожу номер своего студ. билета с

```

kbshabakova@vbox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
kbshabakova@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
kbshabakova@vbox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132242469
Ваш вариант: 10

```

клавиатуры, программа вывела, что мой вариант – 10.

Рис. 22: Запуск исполняемого файла

4.2.1 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
2. `call atoi` используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`

3. 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`
5. Инструкция `inc edx` увеличивает значение регистра `edx` на 1

6. За вывод на экран результатов вычислений отвечают строки:

```

mov eax,edx
call iprintLF

```

4.3 Выполнение заданий для самостоятельной работы

Создаю файл `lab6-4.asm` с помощью утилиты `touch` (рис. 23).

```

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

```

Рис. 23: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $5(x + 18) - 28$ (рис. 24). Это выражение было под вариантом 10.

```

~/.work/arch-pc/lab06/lab6-4.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
add eax, 18; eax = eax + 18 = x + 18
mov ebx, 5
mul ebx; EAX=EBX*EAX = 5*(x+18)
add eax, -28; eax = eax - 28 = 5*(x+18) - 28
mov edi, eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax, rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения

```

Рис.
24:

Написание программы

Создаю и запускаю исполняемый файл (рис. 25). При вводе значения 1, вывод 67.

```

nasm -f elf lab6-4.asm
kbshabakova@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
kbshabakova@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 67kbshabakova@vbox:~/work/arch-pc/lab06$

```

Рис. 25: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. 26). Программа отработала верно.

```

kbshabakova@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 4
Результат: 82kbshabakova@vbox:~/work/arch-pc/lab06$

```

Рис. 26: Запуск исполняемого файла

Листинг 4.1. Программа для вычисления значения выражения $5(x + 18) - 28$.

`%include 'in_out.asm'` ; подключение внешнего файла

SECTION .data

msg: DB 'Введите значение переменной x: ',0

rem: DB 'Результат: ',0

SECTION .bss

x: RESB 80

SECTION .text

GLOBAL _start

_start:

; ---- Вычисление выражения

mov eax, msg

call sprint

mov ecx, x

mov edx, 80

call sread

mov eax,x

call atoi

add eax,18; $eax = eax + 18 = x + 18$

mov ebx,5

*mul ebx; $EAX = EBX * EAX = 5 * (x + 18)$*

*add eax,-28; $eax = eax - 28 = 5 * (x + 18) - 28$*

mov edi,eax ; запись результата вычисления в 'edi'

; ---- Вывод результата на экран

mov eax,rem ; вызов подпрограммы печати

call sprint ; сообщения 'Результат: '

mov eax,edi ; вызов подпрограммы печати значения

call iprint ; из 'edi' в виде символов

call quit ; вызов подпрограммы завершения

5 Выводы

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

6 Список литературы

1. [Лабораторная работа №6](#)

Н
У
Р
Е
Р
L
I
N
K

"

h
t
t
p
s
:
/
/
w
w
w

.

r
a
p
i
d
t
a
b
l
e
s
.
c
o
m
/
c
o
d
e