

# **Шаблон отчёта по лабораторной работе**

**Простейший вариант**

Алади Принц Чисом, Нкабд-05-22

# 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>8</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>10</b>
<b>5</b>	<b>Задание для самостоятельной работы</b>	<b>22</b>
<b>6</b>	<b>Листинги программ:</b>	<b>24</b>
<b>7</b>	<b>Выводы</b>	<b>27</b>
	<b>Список литературы</b>	<b>28</b>

# Список иллюстраций

4.1	4.1 . . . . .	10
4.2	4.2 . . . . .	11
4.3	4.3 . . . . .	11
4.4	4.4 . . . . .	12
4.5	4.5 . . . . .	12
4.6	4.6 . . . . .	12
4.7	4.7 . . . . .	12
4.8	4.8 . . . . .	13
4.9	4.9 . . . . .	13
4.10	4.10 . . . . .	14
4.11	4.11 . . . . .	14
4.12	4.12 . . . . .	15
4.13	4.13 . . . . .	15
4.14	4.14 . . . . .	15
4.15	4.15 . . . . .	16
4.16	4.16 . . . . .	16
4.17	4.17 . . . . .	17
4.18	4.18 . . . . .	17
4.19	4.19 . . . . .	18
4.20	4.20 . . . . .	19
4.21	4.21 . . . . .	20
5.1	4.22 . . . . .	22
5.2	4.22 . . . . .	23

## **Список таблиц**

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

## 2 Задание

1. Создать файл lab7-1.asm и ввести в него программу из листинга 1, создать исполняемый файл и запустить его.

2. Исправить листинг 1, заменив строки

`mov eax, '6'` `mov ebx, '4'` на строки `mov eax, 6` `mov ebx, 4` Создать исполняемый файл и запустить его, пользуясь таблицей ASCII определить какому символу соответствует код 10.

3. Создать файл lab7-2.asm, ввести в него программу из листинга 2, создать исполняемый файл и запустить его.

4. Исправить листинг 2, заменив строки

`mov eax, '6'` `mov ebx, '4'` на строки `mov eax, 6` `mov ebx, 4` Создать исполняемый файл и запустить его.

5. Заменить функцию `iPrintLF` на `iPrint`. Создать исполняемый файл и запустить его. Выяснить чем отличается вывод функций `iPrintLF` и `iPrint`.

6. Создать файл lab7-3.asm, заполнить его соответственно с листингом 3, создать исполняемый файл и запустить его.

7. Изменить файл так, чтобы программа вычисляла выражение  $\frac{4 \times 6 + 2}{5}$

8. Создать файл “вариант”, заполнить его соответственно с листингом 4, создать исполняемый файл и запустить его.

9. Ответить на вопросы по разделу.
10. Написать программу для вычисления выражения  $5 \cdot (x + 18) - 28$  и проверить его при  $x=2$  и при  $x=3$ .

### 3 Теоретическое введение

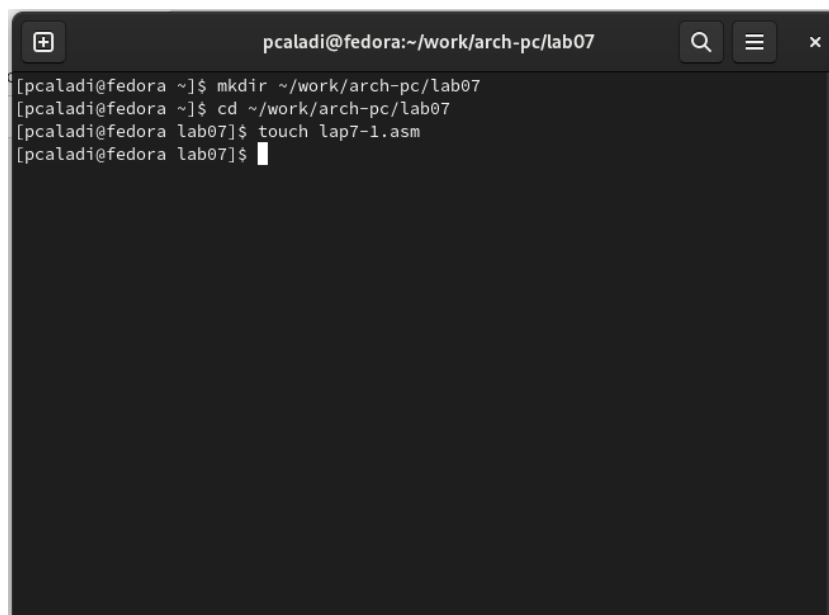
Схема команды целочисленного сложения `add` (от англ. *addition* - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. 1. Команда `add` работает как с числами со знаком, так и без знака и выглядит следующим образом: `add` , Допустимые сочетания операндов для команды `add` аналогичны сочетаниям операндов для команды `mov`. Так, например, команда `add eax,ebx` прибавит значение из регистра `eax` к значению из регистра `ebx` и запишет результат в регистр `eax`. 2. Команда целочисленного вычитания `sub` (от англ. *subtraction* – вычитание) работает аналогично команде `add` и выглядит следующим образом: `sub` , Так, например, команда `sub ebx,5` уменьшает значение регистра `ebx` на 5 и записывает результат в регистр `ebx`. 3. Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: `inc` (от англ. *increment*) и `dec` (от англ. *decrement*), которые увеличивают и уменьшают на 1 свой операнд. Эти команды содержат один операнд и имеет следующий вид: `inc dec` Операндом может быть регистр или ячейка памяти любого размера. Команды инкремента и декремента выгодны тем, что они занимают меньше места, чем соответствующие команды сложения и вычитания. Так, например, команда `inc ebx` увеличивает значение регистра `ebx` на 1, а команда `dec ax` уменьшает значение регистра `ax` на 1. 4. Еще одна команда, которую можно отнести к арифметическим командам это команда изменения знака `neg`: `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. 5. Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производится по-разному, поэтому существуют различные команды. Для беззнакового умножения используется команда `mul` (от англ. `multiply` – умножение): `mul` Для знакового умножения используется команда `imul`: `imul` Для команд умножения один из сомножителей указывается в команде и должен находиться в регистре или в памяти, но не может быть непосредственным операндом. Второй сомножитель в команде явно не указывается и должен находиться в регистре `EAX`, `AX` или `AL`, а результат помещается в регистры `EDX:EAX`, `DX:AX` или `AX`, в зависимости от размера операнда. 6. Для деления, как и для умножения, существует 2 команды `div` (от англ. `divide` – деление) и `idiv`: `div` `idiv` В командах указывается только один операнд – делитель, который может быть регистром или ячейкой памяти, но не может быть непосредственным операндом. Местоположение делимого и результата для команд деления зависит от размера делителя. Кроме того, так как в результате деления получается два числа – частное и остаток, то эти числа помещаются в определённые регистры

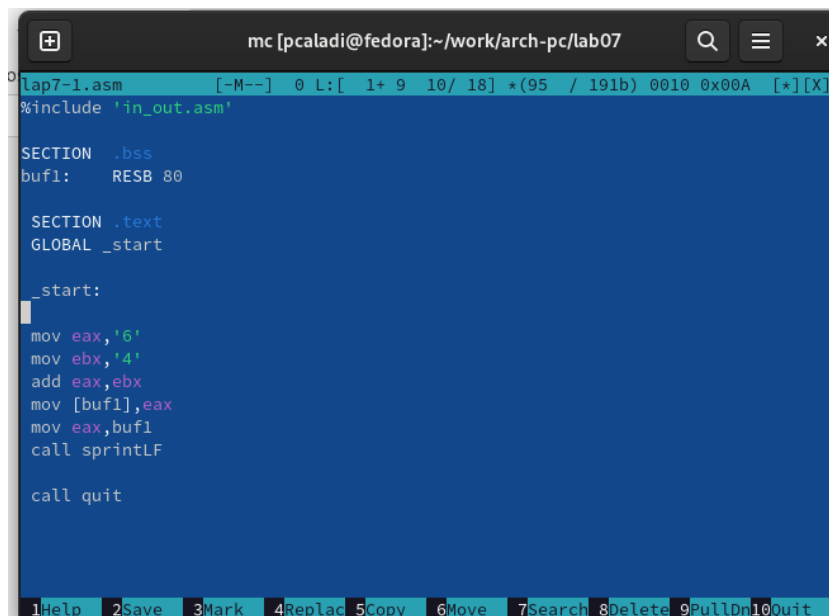
## 4 Выполнение лабораторной работы

1. Я создала файл lab7-1.asm и ввела в него программу из листинга 1, создала исполняемый файл и запустила его(рис. 4.1) (рис. 4.2) (рис. 4.3)



```
pcaladi@fedora:~/work/arch-pc/lab07
[pcaladi@fedora ~]$ mkdir ~/work/arch-pc/lab07
[pcaladi@fedora ~]$ cd ~/work/arch-pc/lab07
[pcaladi@fedora lab07]$ touch lab7-1.asm
[pcaladi@fedora lab07]$
```

Рис. 4.1: 4.1



```
lab7-1.asm [-M--] 0 L: [ 1+ 9 10/ 18] *(95 / 191b) 0010 0x00A [*][X]
#include 'in_out.asm'

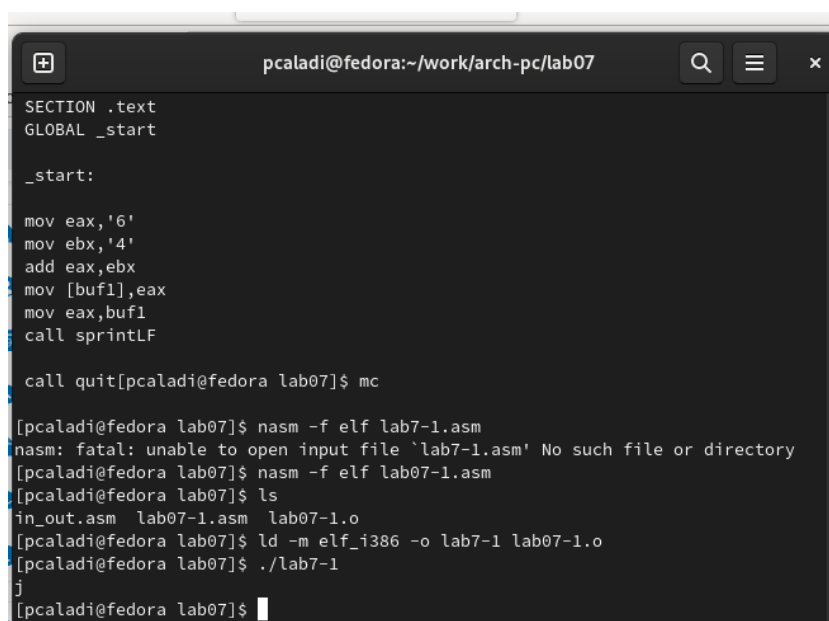
SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start

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

call quit
```

Рис. 4.2: 4.2



```
SECTION .text
GLOBAL _start

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

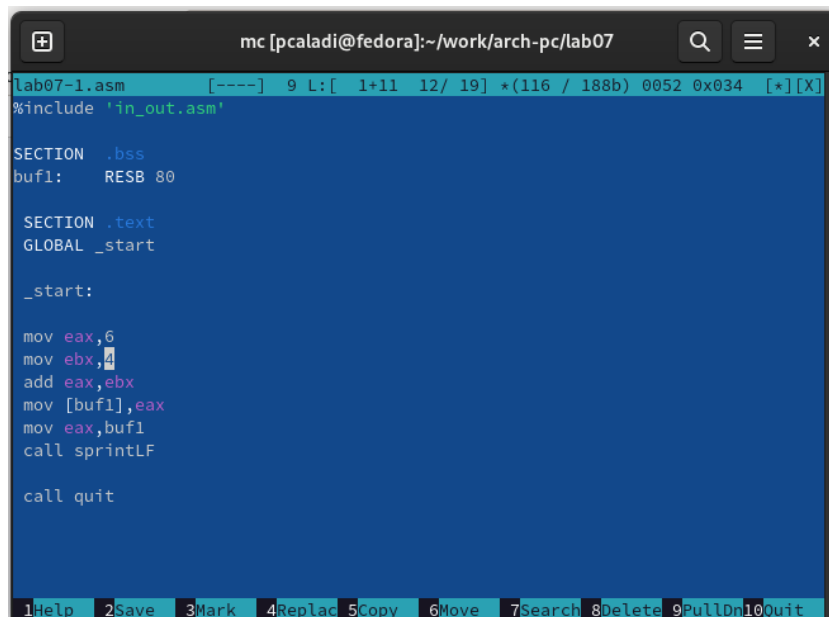
call quit[pcaladi@fedora lab07]$ mc

[pcaladi@fedora lab07]$ nasm -f elf lab7-1.asm
nasm: fatal: unable to open input file `lab7-1.asm' No such file or directory
[pcaladi@fedora lab07]$ nasm -f elf lab07-1.asm
[pcaladi@fedora lab07]$ ls
in_out.asm lab07-1.asm lab07-1.o
[pcaladi@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab07-1.o
[pcaladi@fedora lab07]$ ./lab7-1
j
[pcaladi@fedora lab07]$
```

Рис. 4.3: 4.3

## 2. Исправила листинг 1, заменив строки

mov eax,'6' mov ebx,'4' на строки mov eax,6 mov ebx,4 Создала исполняемый файл и запустила его, пользуясь таблицей ASCII определила какому символу соответствует код 10,(рис. 4.4) (рис. 4.5) (рис. 4.6)



The screenshot shows a text editor window titled 'mc [pcaladi@fedora]:~/work/arch-pc/lab07'. The editor contains assembly code for 'lab07-1.asm'. The code includes a comment line, an include directive for 'in\_out.asm', and two sections: '.bss' with a variable 'buf1' of size 80, and '.text' with a global symbol '\_start'. The main code block starts with '\_start:' and contains instructions to move values into registers, add them, store the result in 'buf1', and call 'sprintf' and 'quit'.

```
lab07-1.asm  [----]  9  L: [ 1+11 12/ 19] *(116 / 188b) 0052 0x034 [*][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
```

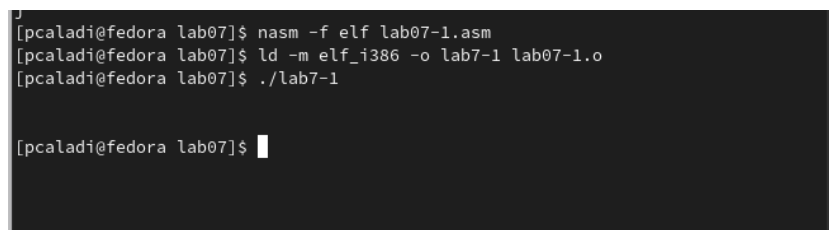
Рис. 4.4: 4.4



The screenshot shows a terminal window with the prompt '[pcaladi@fedora lab07]'. The user enters './lab7-1' and presses enter, resulting in the output 'j'.

```
[pcaladi@fedora lab07]$ ./lab7-1
j
[pcaladi@fedora lab07]$
```

Рис. 4.5: 4.5



The screenshot shows a terminal window with the prompt '[pcaladi@fedora lab07]'. The user enters three commands: 'nasm -f elf lab07-1.asm', 'ld -m elf\_i386 -o lab7-1 lab07-1.o', and './lab7-1'. The output shows the assembly and linking process.

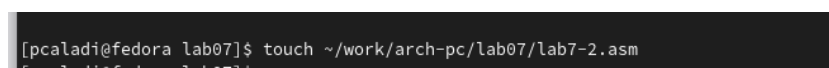
```
[pcaladi@fedora lab07]$ nasm -f elf lab07-1.asm
[pcaladi@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab07-1.o
[pcaladi@fedora lab07]$ ./lab7-1

[pcaladi@fedora lab07]$
```

Рис. 4.6: 4.6

Вывод: код 10 соответствует символу переноса строки, но на экране этот символ не отображается.

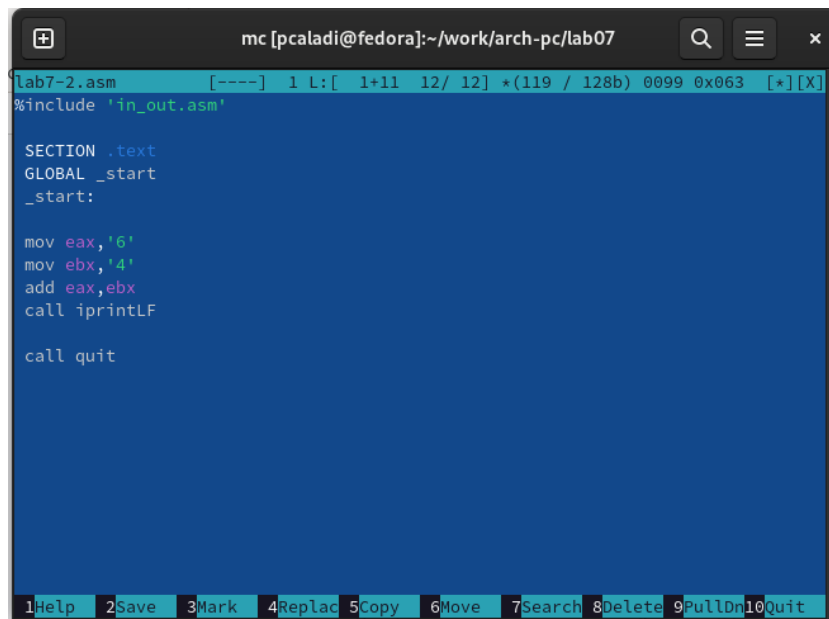
3. Создала файл lab7-2.asm, ввела в него программу из листинга 2, создала исполняемый файл и запустила его (рис. 4.7) (рис. 4.8) (рис. 4.9)



The screenshot shows a terminal window with the prompt '[pcaladi@fedora lab07]'. The user enters the command 'touch ~/work/arch-pc/lab07/lab7-2.asm'.

```
[pcaladi@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-2.asm
[pcaladi@fedora lab07]$
```

Рис. 4.7: 4.7



The screenshot shows a text editor window titled 'mc [pcaladi@fedora]:~/work/arch-pc/lab07'. The editor displays the following assembly code:

```
lab7-2.asm [----] 1 L: [ 1+11 12/ 12] *(119 / 128b) 0099 0x063 [*][X]
#include 'in_out.asm'

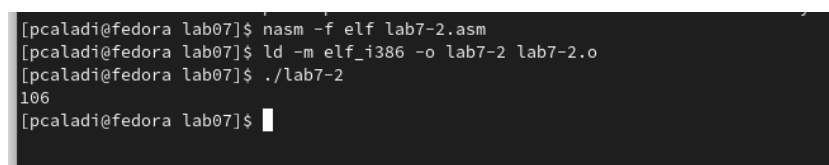
SECTION .text
GLOBAL _start
_start:

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

call quit
```

At the bottom of the editor, there is a toolbar with buttons: 1Help, 2Save, 3Mark, 4Replac, 5Copy, 6Move, 7Search, 8Delete, 9PullDn, 10Quit.

Рис. 4.8: 4.8



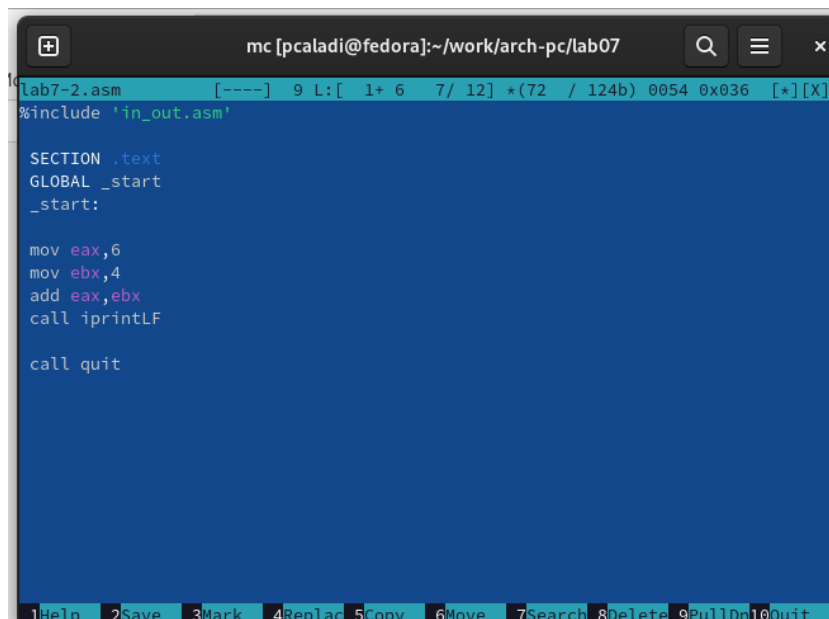
The screenshot shows a terminal window with the following commands and output:

```
[pcaladi@fedora lab07]$ nasm -f elf lab7-2.asm
[pcaladi@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[pcaladi@fedora lab07]$ ./lab7-2
106
[pcaladi@fedora lab07]$
```

Рис. 4.9: 4.9

#### 4. Исправила листинг 2, заменив строки

mov eax,'6' mov ebx,'4' на строки mov eax,6 mov ebx,4 Создала исполняемый файл и запустила его (рис. 4.10) (рис. 4.11)

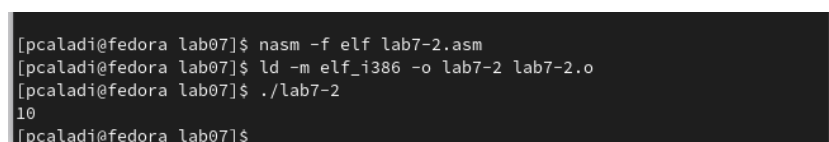


The screenshot shows a code editor window titled 'mc [pcaladi@fedora]:~/work/arch-pc/lab07'. The code in the editor is as follows:

```
lab7-2.asm [----] 9 L: [ 1+ 6 7/ 12] *(72 / 124b) 0054 0x036 [*][X]  
%include 'in_out.asm'  
  
SECTION .text  
GLOBAL _start  
_start:  
  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
  
call quit
```

The editor has a menu bar at the bottom with options: 1.Hello, 2.Save, 3.Mark, 4.Replac, 5.Copy, 6.Move, 7.Search, 8.Delete, 9.PullDn, 10.Quit.

Рис. 4.10: 4.10

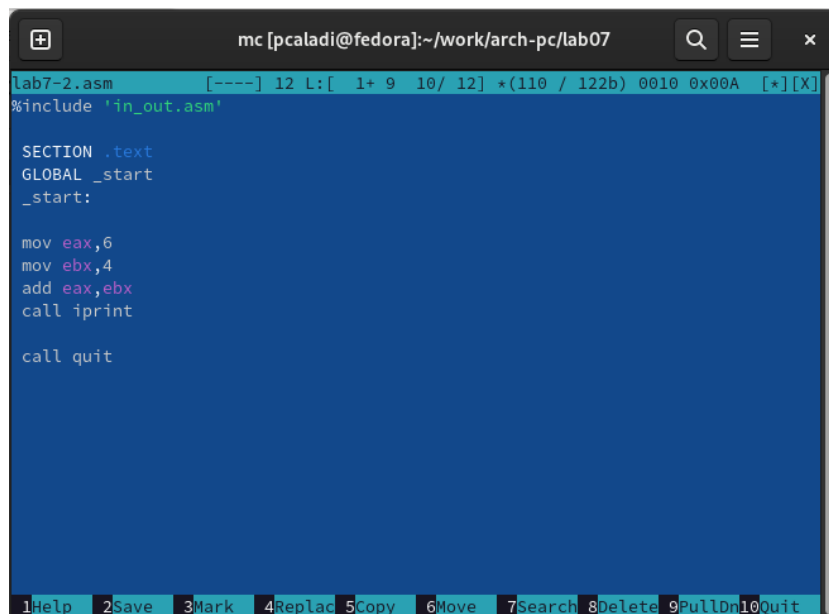


The screenshot shows a terminal window with the following commands and output:

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

Рис. 4.11: 4.11

5. Заменяла функцию `iprintLF` на `iprint`. Создала исполняемый файл и запустила его. Выяснила чем отличается вывод функций `iprintLF` и `iprint` (рис. 4.12) (рис. 4.13)



The screenshot shows a text editor window titled 'mc [pcaladi@fedora]:~/work/arch-pc/lab07'. The editor displays the contents of 'lab7-2.asm'. The code includes a header line with metadata, an include directive for 'in\_out.asm', and assembly instructions for setting up a text section, global symbols, and performing arithmetic and system calls. A menu bar at the bottom lists actions like Help, Save, Mark, Replace, Copy, Move, Search, Delete, PullDown, and Quit.

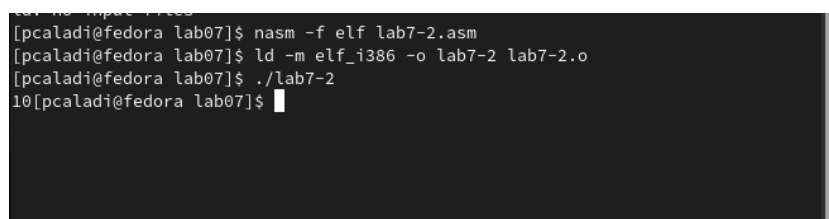
```
lab7-2.asm [----] 12 L: [ 1+ 9 10/ 12] *(110 / 122b) 0010 0x00A [*][X]
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 4.12: 4.12



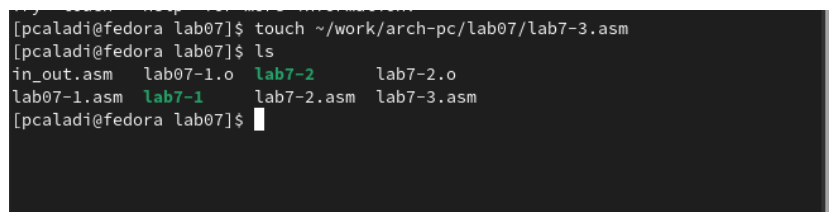
The screenshot shows a terminal window with the following commands and output: 'nasm -f elf lab7-2.asm' to assemble the file, 'ld -m elf\_i386 -o lab7-2 lab7-2.o' to link it into an executable, and './lab7-2' to run it. The prompt returns to the shell after each command.

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

Рис. 4.13: 4.13

Вывод: отличие состоит в том, что iprint не совершает перенос строки.

6. Создала файл lab7-3.asm, заполнила его соответственно с листингом 3, создала исполняемый файл и запустила его (рис. 4.14) (рис. 4.15) (рис. 4.16)



The screenshot shows a terminal window with the following commands and output: 'touch ~/work/arch-pc/lab07/lab7-3.asm' to create a new file, and 'ls' to list the directory contents. The listing shows several files, including 'lab7-1', 'lab7-2', and 'lab7-3.asm'.

```
[pcaladi@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-3.asm
[pcaladi@fedora lab07]$ ls
in_out.asm  lab07-1.o  lab7-2    lab7-2.o
lab07-1.asm lab7-1     lab7-2.asm lab7-3.asm
[pcaladi@fedora lab07]$
```

Рис. 4.14: 4.14

```

mc [pcaladi@fedora]:~/work/arch-pc/lab07
lab7-3.asm [----] 3 L: [ 1+21 22/ 38] *(582 /1444b) 0109 0x06D [*] [X]
;-----
; 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,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. 4.15: 4.15

```

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

```

Рис. 4.16: 4.16

7. Изменила файл так, чтобы программа вычисляла выражение  $\frac{4 \times 186 + 2}{5}$  (рис. 4.17) (рис. 4.18)



```

lab7-3.asm [----] 20 L: [ 12+10 22/ 38] *(599 /1444b) 0010 0x00A [*][X]
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/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 iprintlnLF ; из 'edi' в виде символов

mov eax,rem ; вызов подпрограммы печати

```

Рис. 4.17: 4.17

```

mc [pcaladi@fedora]:~/work/arch-pc/lab07
lab7-3.asm [----] 3 L: [ 1+21 22/ 38] *(582 /1444b) 0109 0x06D [*][X]
;-----
; 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,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 iprintlnLF ; 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 iprintlnLF ; из 'edx' (остаток) в виде символов

call quit ; вызов подпрограммы завершения

```

Рис. 4.18: 4.18

8. Создала файл “вариант”, заполнила его соответственно с листингом 4, создала исполняемый файл и запустила его (рис. 4.19) (рис. 4.20) (рис. 4.21)

```
[pcaladi@fedora lab07]$ touch ~/work/arch-pc/lab07/variant.asm
[pcaladi@fedora lab07]$ ls
in_out.asm  lab07-1.o  lab7-2      lab7-2.o  lab7-3.asm  variant.asm
lab07-1.asm lab7-1     lab7-2.asm  lab7-3    lab7-3.o
[pcaladi@fedora lab07]$
```

Рис. 4.19: 4.19

```
variant.asm [----] 2 L:[ 1+26 27/ 39] *(558 / 676[*])[X]
;-----
; Программа вычисления варианта
;-----

%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`

xor edx, edx
mov ebx, 20
div ebx
inc edx

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

call quit
```

1Help 2Save 3Mark 4Re~ac 5Copy 6Move 7Se~ch 8De~te 9Pu~Dn

Рис. 4.20: 4.20

```
[pcaladi@fedora lab07]$ nasm -f elf variant.asm
[pcaladi@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[pcaladi@fedora lab07]$ ./variant
Введите No студенческого билета:
1032225007
Ваш вариант: 8
[pcaladi@fedora lab07]$
```

Рис. 4.21: 4.21

Выполнив те же вычисления вручную, выяснила, что ответ, данный программой, верен.

#### 9. Отвечаю на вопросы по разделу:

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

```
‘mov eax,rem’ ‘call sprint’
```

2. Для чего используются следующие инструкции? ‘mov esx, x’ - адрес вводимой строки x записывается в регистр esx. ‘mov edx, 80’ - 80 - длина вводимой строки, записана в edx. ‘call sread’ - считывание ввода с клавиатуры.

3. Для чего используется инструкция “call atoi”?

Эта инструкция вызывает программу из файла “in\_out.asm” и преобразует ascii-код символа в целое число и записывает результат в регистр eax.

4. Какие строки листинга 7.4 отвечают за вычисления варианта?

```
xor edx,edx
```

```
mov ebx,20
```

```
div ebx
```

```
inc edx
```

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

В регистр edx

6. Для чего используется инструкция “inc edx”?

Для того, чтобы прибавить к значению edx единицу

7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений?

```
mov eax,edx
```

```
call iprintLF
```

## 5 Задание для самостоятельной работы

1. Написать программу для вычисления выражения  $5 \times (\times + 18) - 28$  и проверить его при  $x=2$  и при  $x=3$  (рис. 5.1) (рис. 5.2)

```
independ-work.asm [----] 14 L: [ 2+16 18/ 44] *(349 / 698b) 0010 0x00A [*][X]
; Программа вычисления варианта
;-----

%include 'in_out.asm'

SECTION .data
msg: DB 'Введите значение x: ',0
rem: DB 'Ваш результат: ',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'

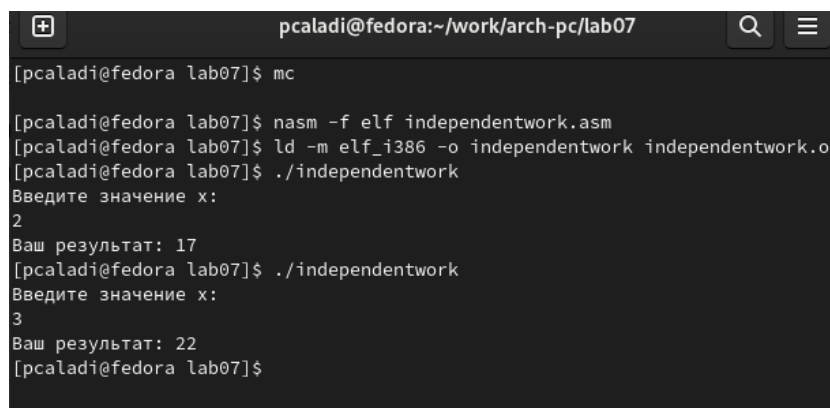
add eax, 18
mov ebx, 5
mul ebx

xor edx, edx
mov ebx, 28
div ebx
inc edx

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

call quit
```

Рис. 5.1: 4.22

A terminal window with a dark background. The title bar shows 'pcaladi@fedora:~/work/arch-pc/lab07'. The terminal content shows a series of commands and their outputs. The user enters 'mc', then 'nasm -f elf independentwork.asm', then 'ld -m elf\_i386 -o independentwork independentwork.o', and finally './independentwork'. The program prompts for input 'x' twice, with values 2 and 3 being entered. The corresponding outputs are 17 and 22.

```
[pcaladi@fedora lab07]$ mc
[pcaladi@fedora lab07]$ nasm -f elf independentwork.asm
[pcaladi@fedora lab07]$ ld -m elf_i386 -o independentwork independentwork.o
[pcaladi@fedora lab07]$ ./independentwork
Введите значение x:
2
Ваш результат: 17
[pcaladi@fedora lab07]$ ./independentwork
Введите значение x:
3
Ваш результат: 22
[pcaladi@fedora lab07]$
```

Рис. 5.2: 4.22

Проверила себя, выполнив вычисления вручную - ответ получен верный.

## 6 Листинги программ:

### 1. lab7-1.asm

```
%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
```

### 2. lab7-2.asm

```
%include 'in_out.asm'

SECTION .text GLOBAL _start _start:

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

### 3. lab7-3.asm

```
;----- ; 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 подпро-
граммы 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
```

#### 4. variant.asm

```
;----- ; Программа вычисления варианта ;-----
%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

xor edx,edx mov ebx,20 div ebx inc edx

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

call quit
```

#### 5. independentwork.asm - самостоятельная работа

```
;----- ; Программа вычисления функции ;-----
%include 'in_out.asm'
```

```

SECTION .data msg: DB 'Введите значение x:',0 rem: DB 'Ваш результат:',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

add eax, 18 mov ebx,5 mul ebx

xor edx,edx mov ebx, 28 neg ebx add eax, ebx

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

call quit

```

## 7 Выводы

В ходе этой лабораторной работы я освоила арифметические инструкции языка ассемблера NASM

# Список литературы

1. Текстовый документ “Лабораторная работа №7. Арифметические операции в NASM.” ::: {#refs} :::