

```
---
## Front matter
title: "Шаблон отчёта по лабораторной работе"
subtitle: "Простейший вариант"
author: "Ариоке Габриэль .О."
group: НКАбд-05-22

## Generic otions
lang: ru-RU
toc-title: "Содержание"

## Bibliography
bibliography: bib/cite.bib
csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

## Pdf output format
toc: true # Table of contents
toc-depth: 2
lof: true # List of figures
lot: true # List of tables
fontsize: 12pt
linestretch: 1.5
papersize: a4
documentclass: scrreprt
## I18n polyglossia
polyglossia-lang:
  name: russian
  options:
    - spelling=modern
    - babelshorthands=true
polyglossia-otherlangs:
  name: english
## I18n babel
babel-lang: russian
babel-otherlangs: english
## Fonts
mainfont: PT Serif
romanfont: PT Serif
sansfont: PT Sans
monofont: PT Mono
mainfontoptions: Ligatures=TeX
romanfontoptions: Ligatures=TeX
sansfontoptions: Ligatures=TeX,Scale=MatchLowercase
monofontoptions: Scale=MatchLowercase,Scale=0.9
## Biblatex
biblatex: true
biblio-style: "gost-numeric"
biblatexoptions:
  - parenttracker=true
  - backend=biber
  - hyperref=auto
  - language=auto
  - autolang=other*
  - citestyle=gost-numeric
```

```
## Pandoc-crossref LaTeX customization
figureTitle: "Рис."
tableTitle: "Таблица"
listingTitle: "Листинг"
lofTitle: "Список иллюстраций"
lotTitle: "Список таблиц"
lolTitle: "Листинги"
## Misc options
indent: true
header-includes:
- \usepackage{indentfirst}
- \usepackage{float} # keep figures where there are in the text
- \floatplacement{figure}{H} # keep figures where there are in the text
---
```

## Содержание

1. Цель работы
2. Задание
3. Теоретическое введение
4. Выполнение лабораторной работы
5. Задание для самостоятельной работы
6. Листинги программ:
7. Выводы

## Цель работы

Освоение арифметических инструкций языка ассемблера NASM

## 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  $\textcircled{C}[\textcircled{R}] = (4 @ 6 + 2) / 5$
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  $5 \textcircled{P} (\textcircled{C} + 18) - 28$  и проверить его при  $x=2$  и при  $x=3$ .

# 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 `ebx` к значению из регистра `eax` 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.

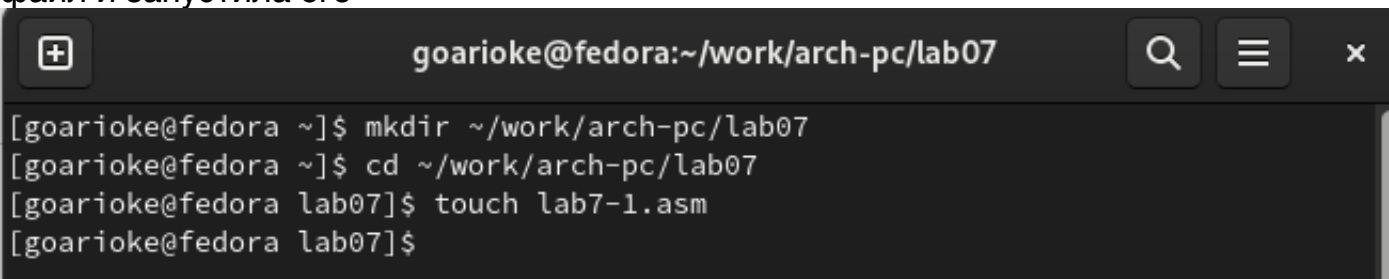
Эти команды содержат один операнд и имеет следующий вид: `inc dec` Операндом может быть регистр или ячейка памяти любого размера. Команды инкремента и декремента выгодны тем, что они занимают меньше места, чем соответствующие команды сложения и вычитания. Так, например, команда `inc ebx` увеличивает значение регистра `ebx` на 1. а команда `dec eax` уменьшает значение регистра `eax` 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 производятся 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

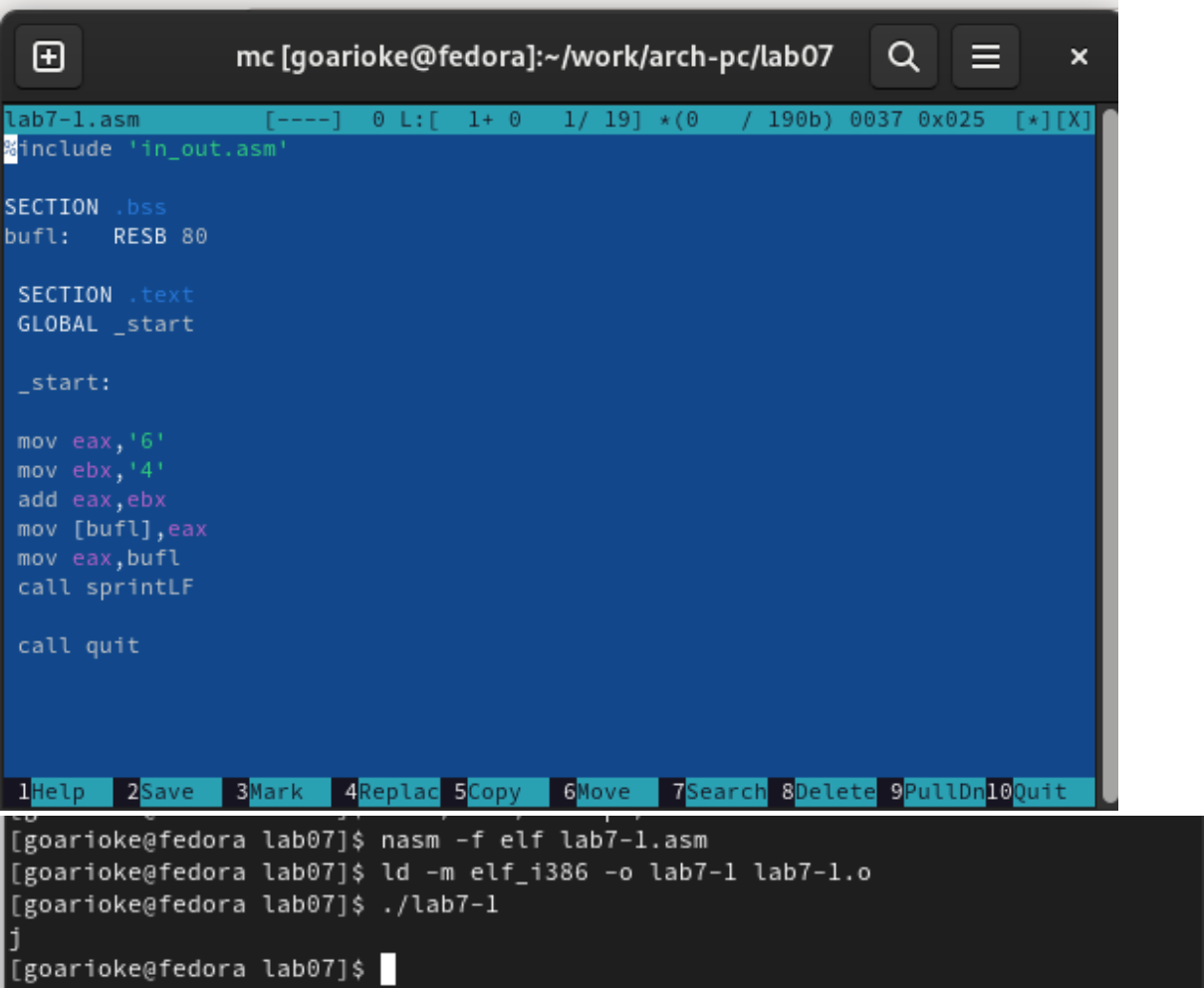
`AL`, а результат помещается в регистры `EDX:EAX`, `DX:AX` или `AX`, в зависимости от размера операнда. 6. Для деления, как и для умножения, существует 2 команды `div` (от англ. *divide* - деление) и `idiv`: `div idiv` В командах указывается только один операнд - делитель, который может быть регистром 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 определенные регистры.

## Выполнение лабораторной работы

1. Я создала файл `lab7-1.asm` и ввела в него программу из листинга 1, создала исполняемый файл и запустила его



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



The image shows a code editor window titled "mc [goarioke@fedora]:~/work/arch-pc/lab07". The editor contains assembly code for "lab7-1.asm". The code includes a header section, a .bss section for a buffer, and a .text section starting with a global \_start label. The main logic moves the ASCII values '6' and '4' into registers, adds them, and prints the result using sprintf. A menu bar at the bottom lists actions: 1Help, 2Save, 3Mark, 4Replac, 5Copy, 6Move, 7Search, 8Delete, 9PullDn, 10Quit.

```
lab7-1.asm [----] 0 L:[ 1+ 0 1/ 19] *(0 / 190b) 0037 0x025 [*][X]
#include 'in_out.asm'

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start

_start:

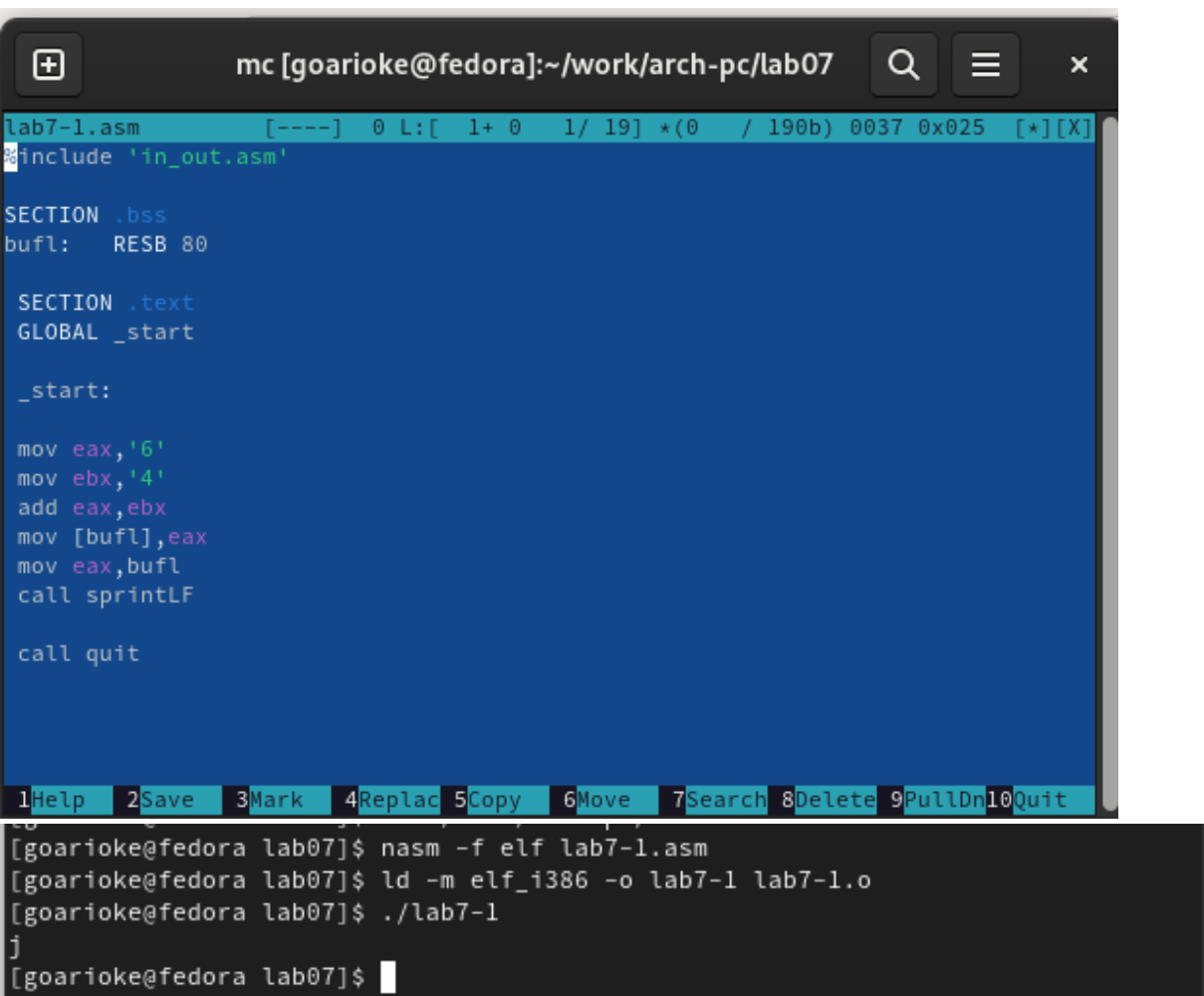
mov eax,'6'
mov ebx,'4'
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintf

call quit
```

Terminal output:

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

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.



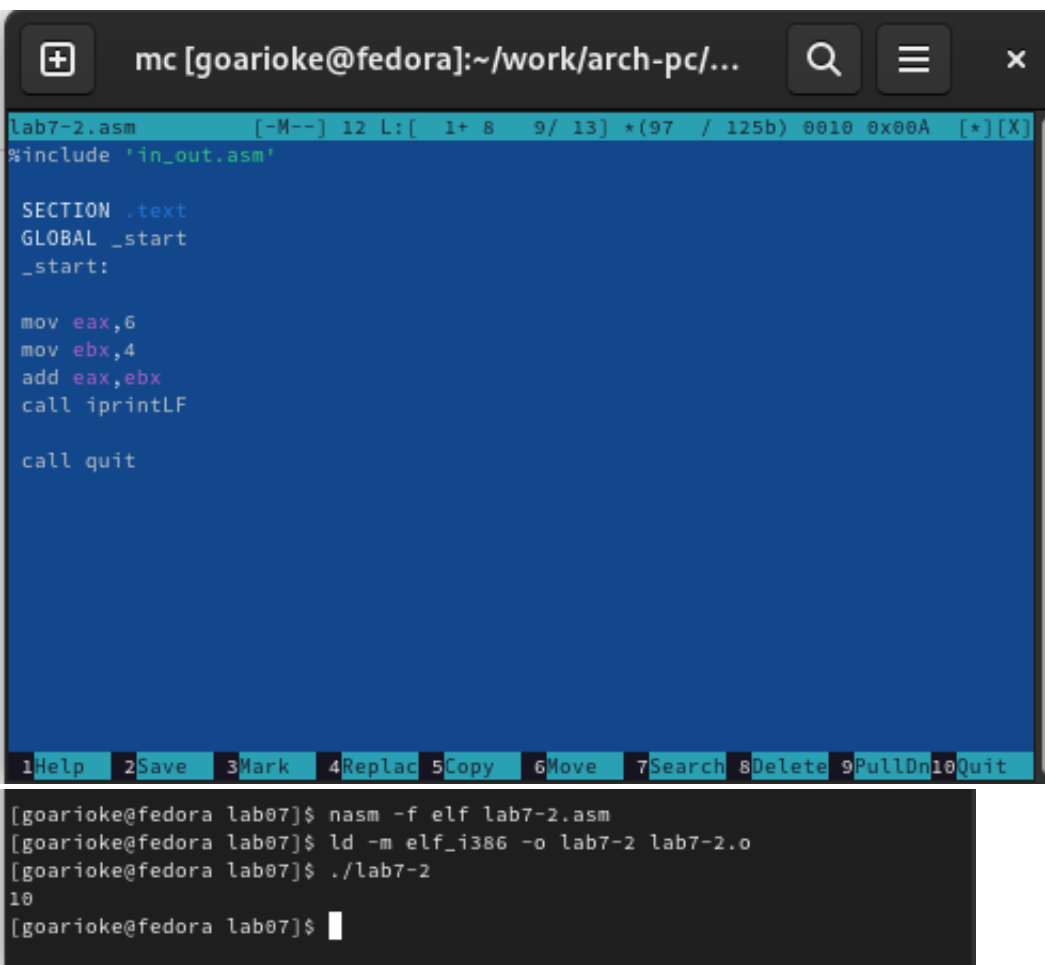
This screenshot is identical to the one above, showing the same assembly code and terminal execution. The terminal output shows a carriage return character (ASCII 10) being printed, which appears as a blank line in the terminal window.

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

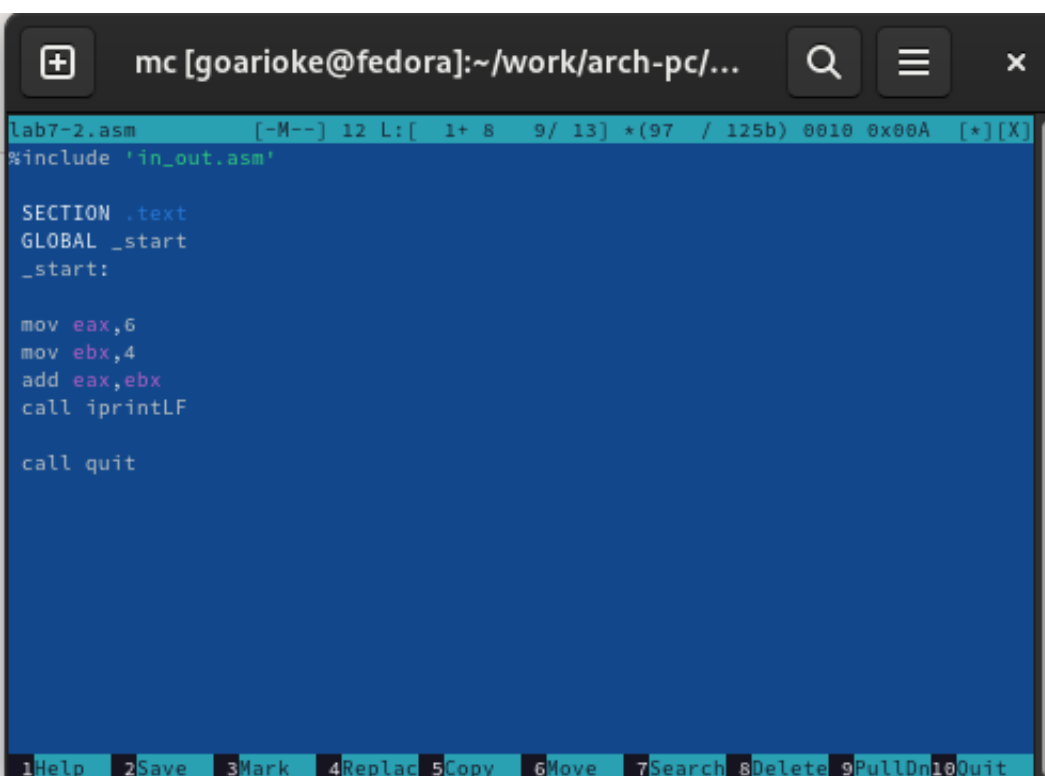
```
[goarioke@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-2.asm  
[goarioke@fedora lab07]$
```



```
lab7-2.asm [-M--] 12 L:[ 1+ 8 9/ 13] *(97 / 125b) 0010 0x00A [*][X]  
%include 'in_out.asm'  
  
SECTION .text  
GLOBAL _start  
_start:  
  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
  
call quit
```

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

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



```
lab7-2.asm [-M--] 12 L:[ 1+ 8 9/ 13] *(97 / 125b) 0010 0x00A [*][X]  
%include 'in_out.asm'  
  
SECTION .text  
GLOBAL _start  
_start:  
  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
  
call quit
```

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

5. Заменяла функцию `iprintLF` на `iprint`. Создала исполняемый файл и запустила его. Выяснила чем отличается вывод функций `iprintLF` и `iprint`

```
lab7-2.asm [-M--] 12 L:[ 1+ 8 9/ 13] *(97 / 125b) 0010 0x00A [*][X]
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

call quit

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit

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

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

```
lab7-3.asm [----] 33 L:[ 1+ 0 1/ 39] *(33 /1445b) 0010 0x00A [*][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,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 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

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

```
[goarioke@fedora lab07]$ nasm -f elf lab7-3.asm
[goarioke@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[goarioke@fedora lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[goarioke@fedora lab07]$
```

7. Изменила файл так, чтобы программа вычисляла выражение  $(9) = (4 @ 186 + 2)/5$

```
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 iprintLF ; из 'edi' в виде символов

mov eax,rem ; вызов подпрограммы печати
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

8. Создала файл "вариант", заполнила его соответственно с листингом 4, со-здала исполняемый файл и запустила его

```
[goarioke@fedora lab07]$ touch ~/work/arch-pc/lab07/variant.asm
[goarioke@fedora 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
[goarioke@fedora lab07]$
```

```

independ~work.asm  [-----]  0 L:[ 1+ 0 1/ 43] *(0 / 697b) 0059 0x03B [*][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

```

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit

```

[goarioke@fedora lab07]$ nasm -f elf variant.asm
[goarioke@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[goarioke@fedora lab07]$ ./variant
Введите No студенческого билета:
1032225081
Ваш вариант: 2
[goarioke@fedora lab07]$

```

Выполнив те же вычисления вручную, выяснила, что ответ, данный програм-мои, 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 ecx, x' - адрес вводимой строки записывается в регистр ecx. 'mov edx, 80' - 80 - длина вводимой строки, записана в ecx. 'call sread' - считывание ввода с клавиатуры.
3. Для чего используется инструкция "call atoi"?  
Эта инструкция вызывает программу из файла "in\_out.asm" и преобразует асци-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"?
6. Для чего используется инструкция "inc edx"?  
Для того, чтобы прибавить к значениям единицу
7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений?  
mov eax,edx  
call iprintLF

## Задание для самостоятельной работы



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

```
lab7-3.asm [----] 33 L:[ 1+ 0 1/ 39] *(33 /1445b) 0010 0x00A [*][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,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 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

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

```
[goarioke@fedora lab07]$ nasm -f elf independentwork.asm
[goarioke@fedora lab07]$ ld -m elf_i386 -o independentwork independentwork.o
[goarioke@fedora lab07]$ ./independentwork
```

Введите значение x:

3

Ваш результат: 22

```
[goarioke@fedora lab07]$
```

```
[goarioke@fedora lab07]$ ./independentwork
```

Введите значение x:

2

Ваш результат: 17

```
[goarioke@fedora lab07]$
```

Проверила себя, выполнив вычисления вручную - ответ получен верный.

## 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 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
```

#### 4. variant.asm

-; Программа вычисления варианта ;

```
%include 'in_out.asm'
```

```
SECTION .data msg: DB 'Введите No студенческого билета:',0 get: 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, ed call iprintLF call quit
```

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

-; Программа вычисления функции

```
%include 'in_out.asm'
```

```
SECTION .data msg: DB 'Введите значение x:',0 em: 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
```

```
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
```

## Выводы

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

## Список литературы

1. Текстовый документ "Лабораторная работа No7. Арифметические операции в NASM." .: (#refs) .: