

Шаблон отчёта по лабораторной работе

Простейший вариант

Ариоке Габриэль Одафе, Нкабд-05-22

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	10
5	Задание для самостоятельной работы	22
6	Листинги программ:	24
7	Выводы	27
	Список литературы	28

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 $\diamond(\diamond) = (4 \diamond 6 + 2)/5$

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

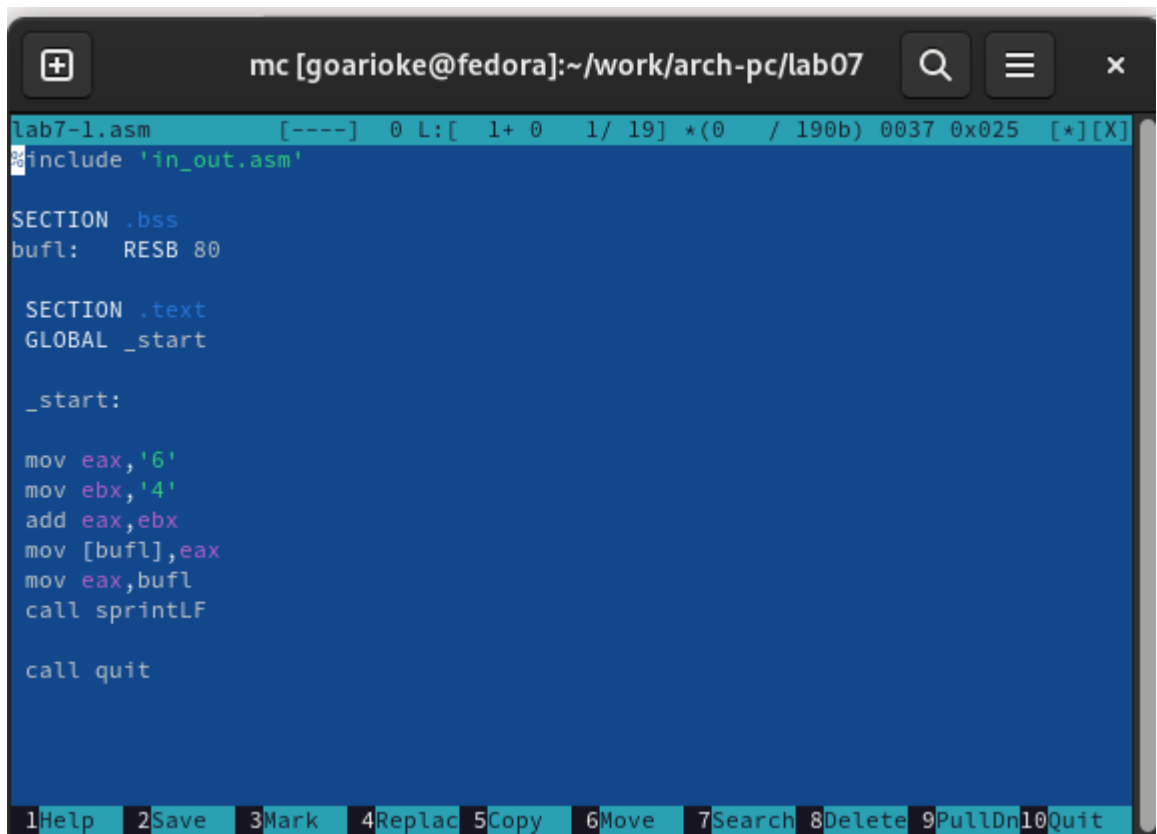
4 Выполнение лабораторной работы

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)

A terminal window with a dark background. The title bar shows the user 'goarioke' on a 'fedora' machine, in the directory '~/work/arch-pc/lab07'. There are search, menu, and close buttons on the right. The terminal text shows the following commands and their outputs:

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

Рис. 4.1: 4.1



```
mc [goarioke@fedora]:~/work/arch-pc/lab07
lab7-1.asm [----] 0 L: [ 1+ 0 1/ 19] *(0 / 190b) 0037 0x025 [*][X]
#include 'in_out.asm'

SECTION .bss
buf1: RESB 80

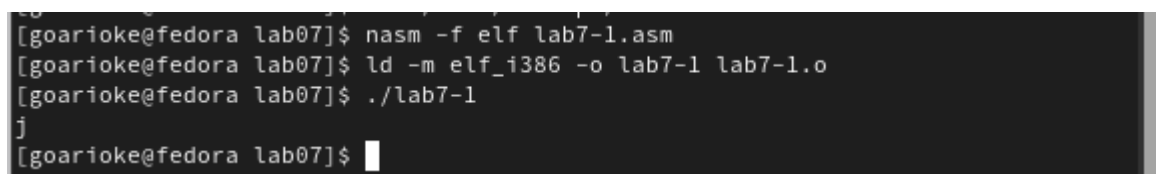
SECTION .text
GLOBAL _start

_start:

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

call _exit
```

Рис. 4.2: 4.2

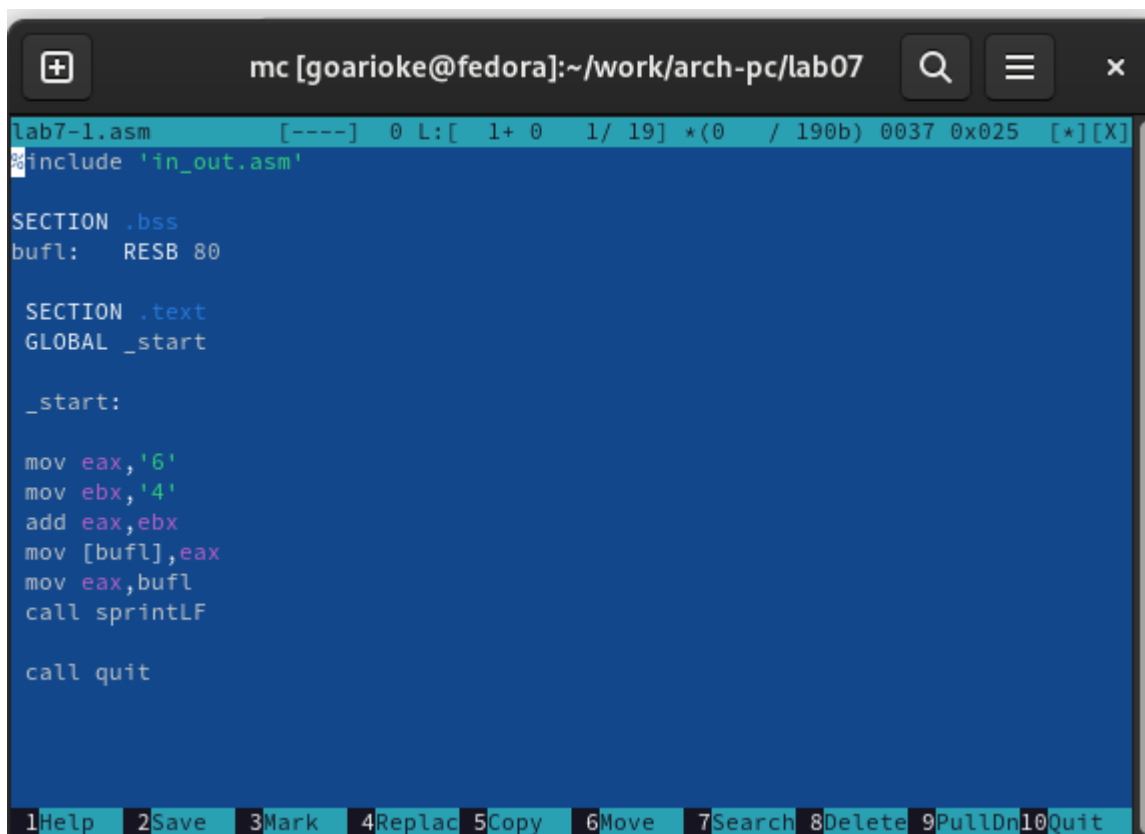


```
[goarioke@fedora lab07]$ nasm -f elf lab7-1.asm
[goarioke@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[goarioke@fedora lab07]$ ./lab7-1
j
[goarioke@fedora lab07]$
```

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)



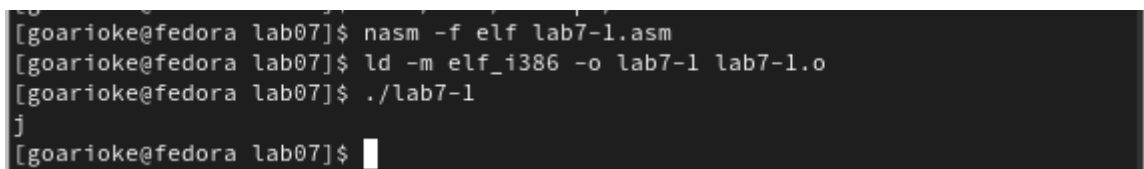
```
lab7-1.asm  [----]  0  L:[  1+ 0  1/ 19]  *(0  / 190b) 0037 0x025  [*][X]
#include 'in_out.asm'

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
```



```
[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]$
```

Рис. 4.6: 4.6

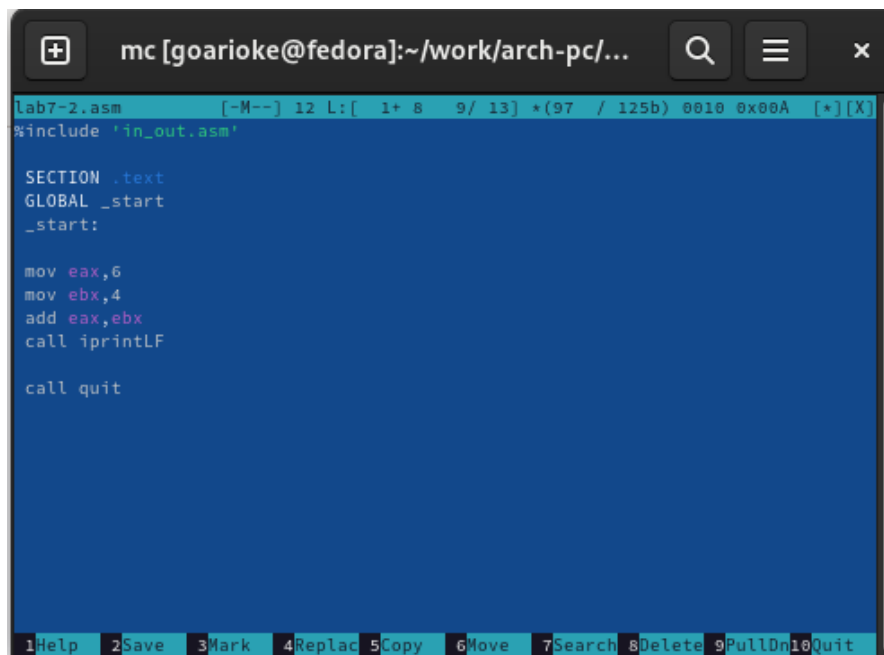
Вывод: код 10 соответствует символу переноса строки, но на экране этот символ не отображается.

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)



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

Рис. 4.7: 4.7



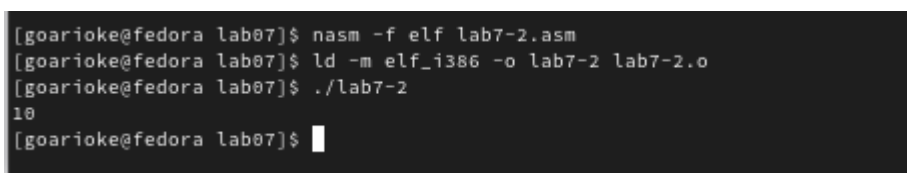
```
lab7-2.asm [-M--] 12 L: [ 1+ 8 9/ 13] *(97 / 125b) 0010 0x00A [*] [X]
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 4.8: 4.8

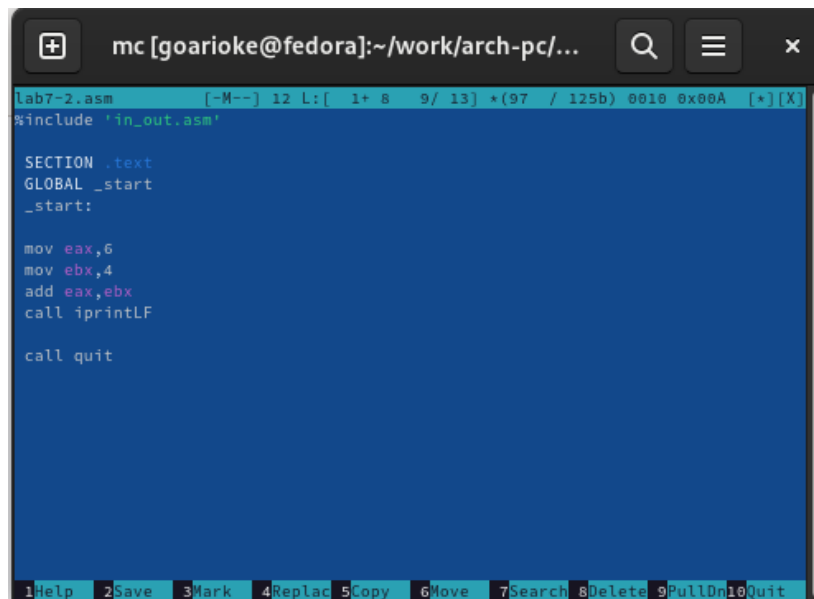


```
[goarioke@fedora lab07]$ nasm -f elf lab7-2.asm
[goarioke@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[goarioke@fedora lab07]$ ./lab7-2
10
[goarioke@fedora lab07]$
```

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)

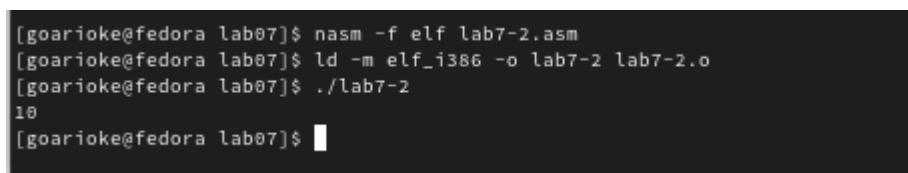


The screenshot shows a text editor window titled 'mc [goarioke@fedora]:~/work/arch-pc/...'. The editor displays the following assembly code:

```
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
```

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

Рис. 4.10: 4.10

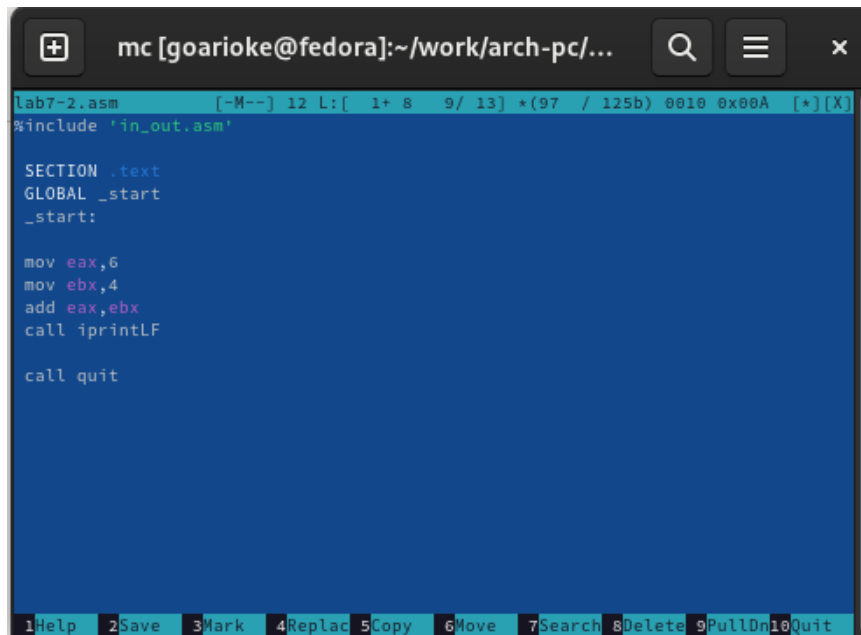


The screenshot shows a terminal window with the following commands and output:

```
[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]$
```

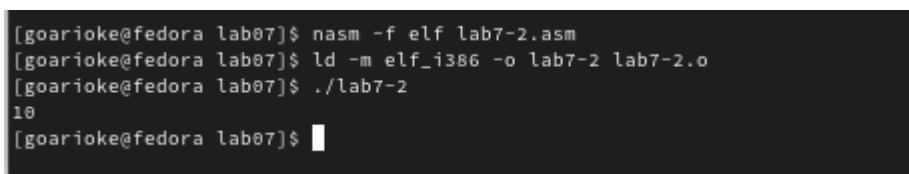
Рис. 4.11: 4.11

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)

A screenshot of a code editor window titled 'mc [goarioke@fedora]:~/work/arch-pc/...'. The editor displays assembly code for 'lab7-2.asm'. The code includes a header line with metadata, an include directive for 'in_out.asm', and a section definition for '.text'. It then defines a global symbol '_start' and contains several instructions: 'mov eax,6', 'mov ebx,4', 'add eax,ebx', 'call iprintLF', and 'call quit'. The editor has a dark theme and a toolbar at the bottom with buttons for Help, Save, Mark, Replace, Copy, Move, Search, Delete, Pull Down, and Quit.

```
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
```

Рис. 4.12: 4.12

A screenshot of a terminal window showing the compilation of the assembly file 'lab7-2.asm'. The user runs 'nasm -f elf lab7-2.asm' to create 'lab7-2.o', then 'ld -m elf_i386 -o lab7-2 lab7-2.o' to create the executable 'lab7-2', and finally './lab7-2' to run it. The output shows the number '10'.

```
[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]$
```

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

```

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

```

Рис. 4.15: 4.15

```

[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]$

```

Рис. 4.16: 4.16

7. Изменила файл так, чтобы программа вычисляла выражение $\diamond(\diamond) = (4 \diamond 186 + 2)/5$ (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 iprintLF ; из 'edi' в виде символов

mov eax,rem ; вызов подпрограммы печати

```

Рис. 4.17: 4.17

```

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

```

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)

```
[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
```

Рис. 4.20: 4.20

```

[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]$

```

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 ecx, x' - адрес вводимой строки записывается в регистр ecx. '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 запишет 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 регистре 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 $5 \diamond (\diamond + 18) - 28$ 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

```
[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]$ █
```

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} :::