

Шаблон отчёта по лабораторной работе

Простейший вариант

Герра Гарсия Паола Валентина, Нкабд-05-22

Содержание

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 $\frac{4 * 6 + 2}{5}$
8. Создать файл "вариант", заполнить его соответственно с листингом 4, создать исполняемый файл и запустить его.
9. Ответить на вопросы по разделу.

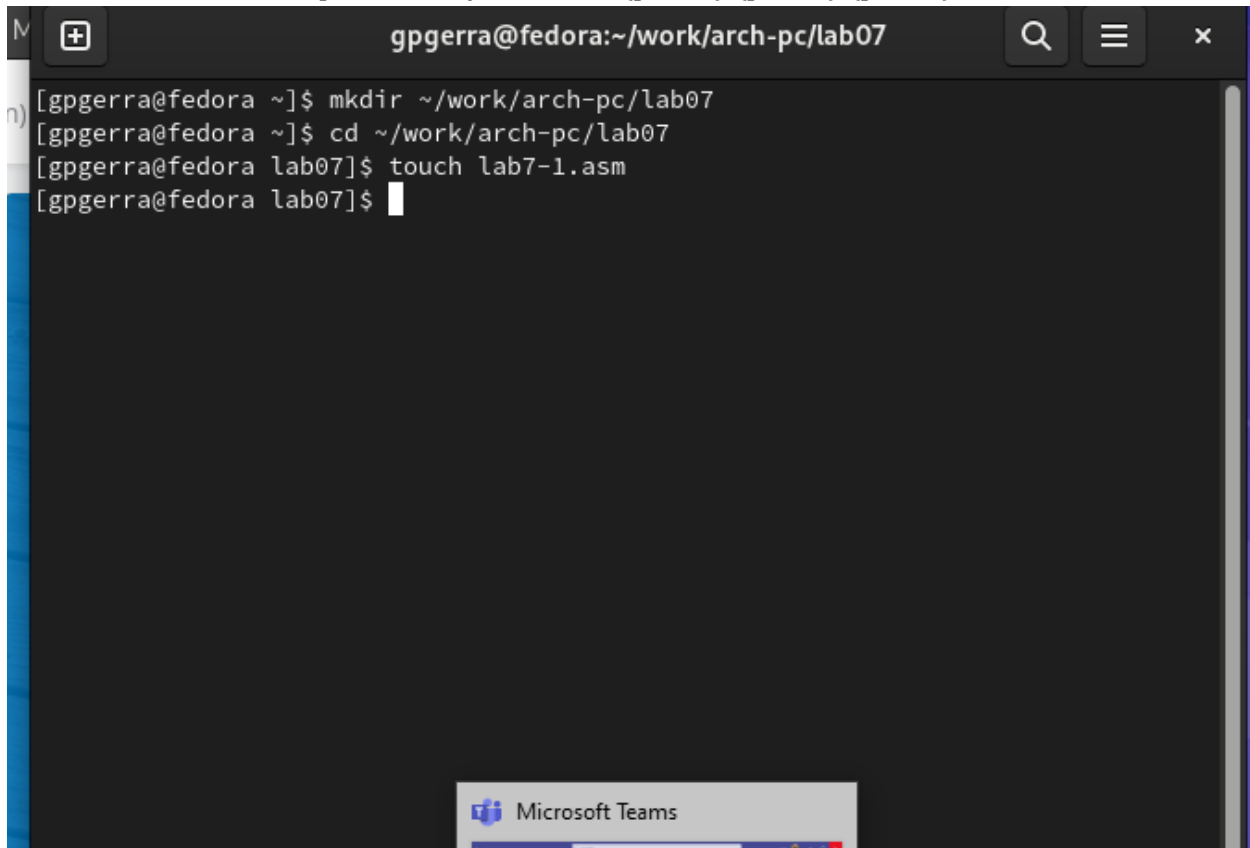
10. Написать программу для вычисления выражения $5 * (x + 18) - 28$ и проверить его при $x=2$ и при $x=3$.

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 имеют 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 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. Вторым сомножителем 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. 1) (рис. 2) (рис. 3)

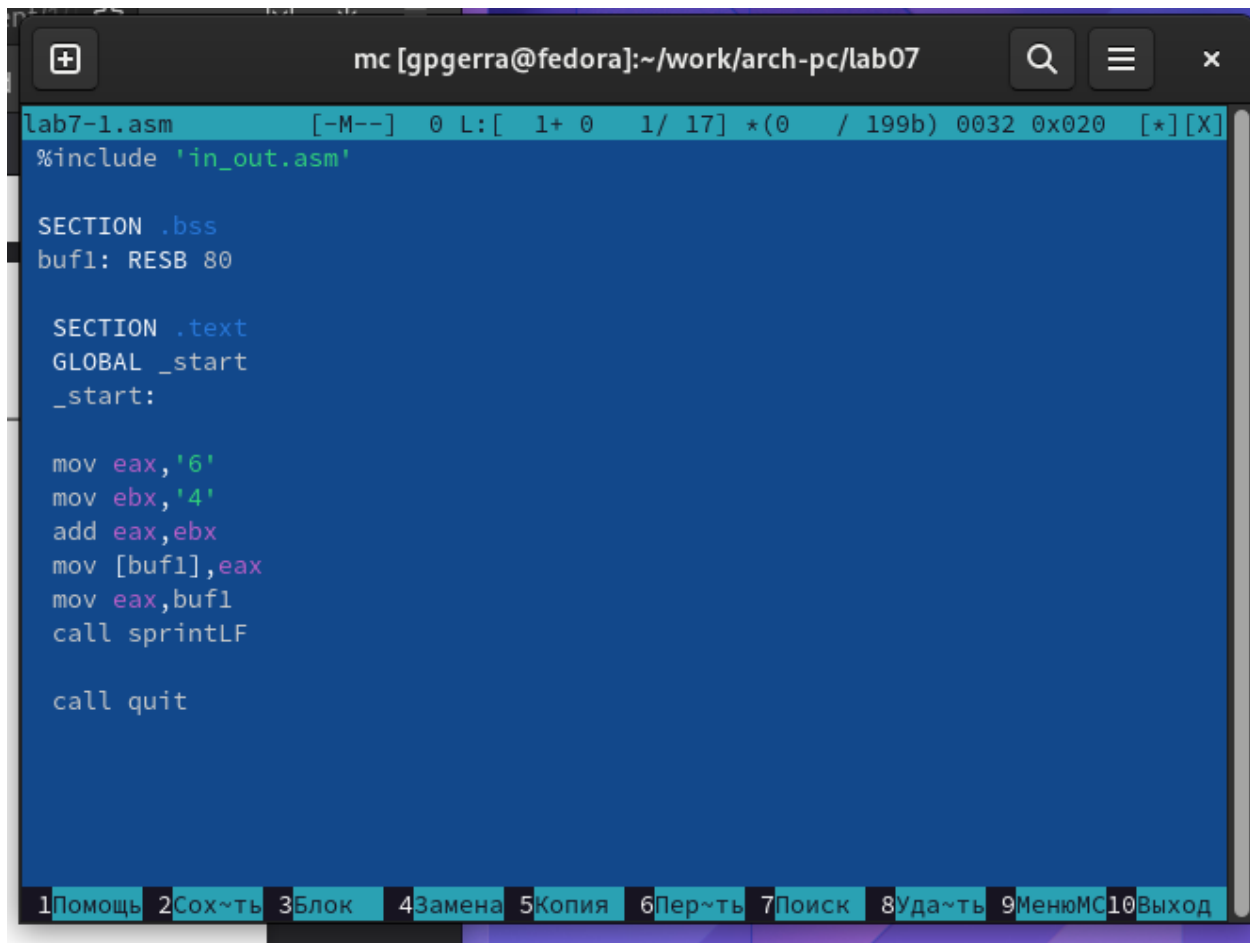


A terminal window titled "gpgerra@fedora:~/work/arch-pc/lab07" with search, menu, and close buttons. The terminal shows the following commands and output:

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

A Microsoft Teams window is visible in the background at the bottom of the terminal.

Рис. 1: 4.1



```
lab7-1.asm [-M--] 0 L: [ 1+ 0 1/ 17] *(0 / 199b) 0032 0x020 [*] [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 quit
```

1Помощь 2Сох~ть 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 2: 4.2

Рис. 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) (рис. 5) (рис. 6)

The screenshot shows a code editor window titled "mc [gpgerra@fedora]:~/work/arch-pc/lab07". The editor displays the file "/home/gpgerra/work/arch-pc/lab07/lab7-1.asm" at 100% zoom. The code is as follows:

```
%include 'in_out.asm'

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

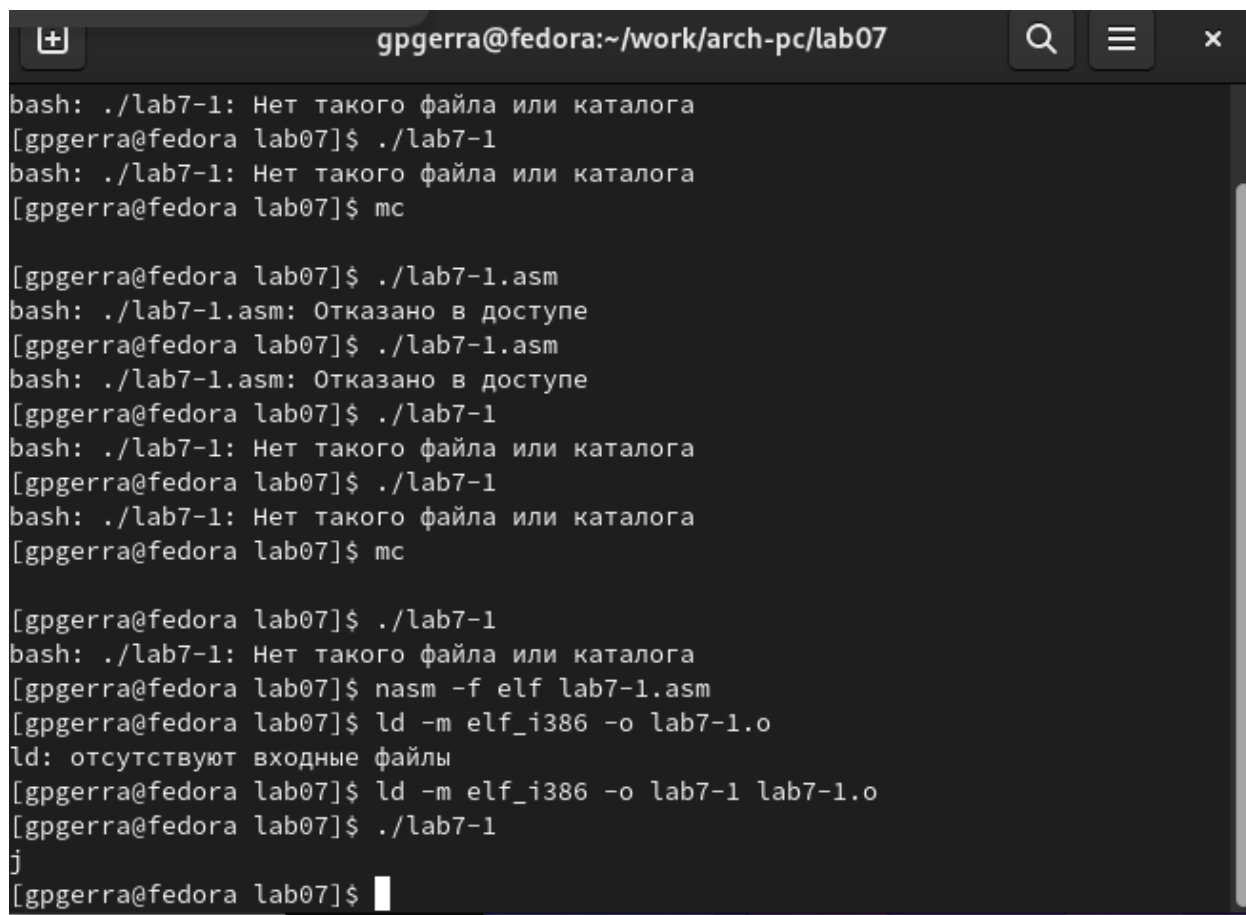
mov eax,'6'
mov ebx,'4'
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintLF

call quit
```

At the bottom of the editor, there is a menu bar with the following items: 1Помощь, 2Раз~рн, 3Выход, 4Нех, 5Пер~ти, 6, 7Поиск, 8Исх~ый, 9Формат, 10Выход.

Рис. 4: 4.4

Рис. 5: 4.5

A terminal window titled 'gpgerra@fedora:~/work/arch-pc/lab07' with search, menu, and close icons. The terminal shows a series of commands and error messages. The user attempts to run './lab7-1', './lab7-1.asm', and './lab7-1' multiple times, receiving 'Нет такого файла или каталога' (No such file or directory) and 'Отказано в доступе' (Access denied) errors. They then use 'mc' (micro) to edit the file. After editing, they run 'nasm -f elf lab7-1.asm' successfully. Then they run 'ld -m elf_i386 -o lab7-1.o', which fails with 'ld: отсутствуют входные файлы' (ld: no input files). Finally, they run 'ld -m elf_i386 -o lab7-1 lab7-1.o', which also fails with the same error. The terminal ends with a carriage return character '\n' being entered, which is not visible on the screen.

```
gpgerra@fedora:~/work/arch-pc/lab07
bash: ./lab7-1: Нет такого файла или каталога
[gpgerra@fedora lab07]$ ./lab7-1
bash: ./lab7-1: Нет такого файла или каталога
[gpgerra@fedora lab07]$ mc

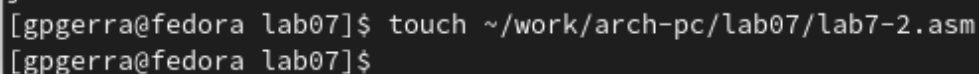
[gpgerra@fedora lab07]$ ./lab7-1.asm
bash: ./lab7-1.asm: Отказано в доступе
[gpgerra@fedora lab07]$ ./lab7-1.asm
bash: ./lab7-1.asm: Отказано в доступе
[gpgerra@fedora lab07]$ ./lab7-1
bash: ./lab7-1: Нет такого файла или каталога
[gpgerra@fedora lab07]$ ./lab7-1
bash: ./lab7-1: Нет такого файла или каталога
[gpgerra@fedora lab07]$ mc

[gpgerra@fedora lab07]$ ./lab7-1
bash: ./lab7-1: Нет такого файла или каталога
[gpgerra@fedora lab07]$ nasm -f elf lab7-1.asm
[gpgerra@fedora lab07]$ ld -m elf_i386 -o lab7-1.o
ld: отсутствуют входные файлы
[gpgerra@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[gpgerra@fedora lab07]$ ./lab7-1
j
[gpgerra@fedora lab07]$ \n
```

Рис. 6: 4.6

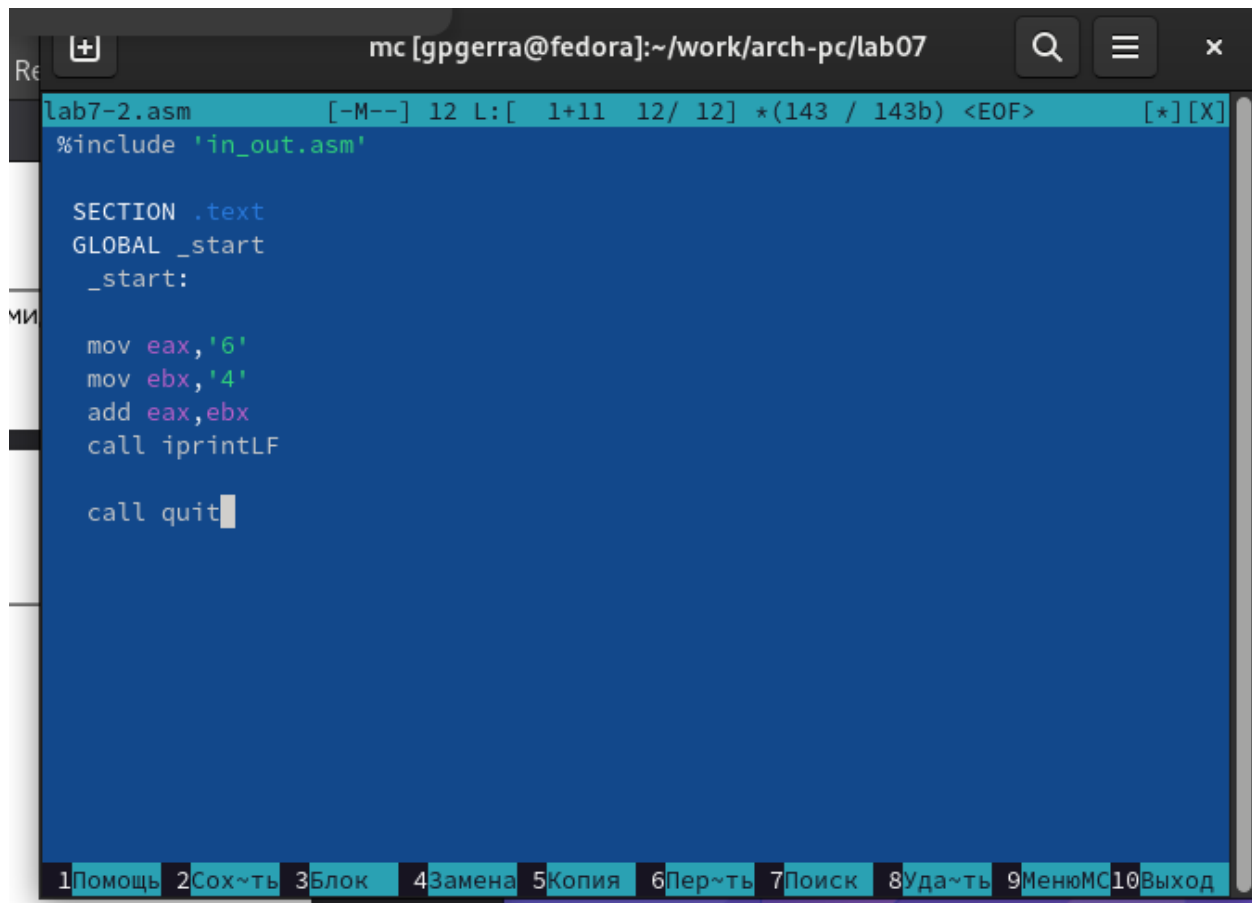
Вывод: код 10 соответствует символу переноса строки, но на экране этот символ не отображается.

3. Создала файл lab7-2.asm, ввела в него программу из листинга 2, создала исполняемый файл и запустила его (рис. 7) (рис. 8) (рис. 9)

A small terminal snippet showing the command to create a new file.

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

Рис. 7: 4.7



```
lab7-2.asm [-M--] 12 L: [ 1+11 12/ 12] *(143 / 143b) <EOF> [*] [X]
#include 'in_out.asm'

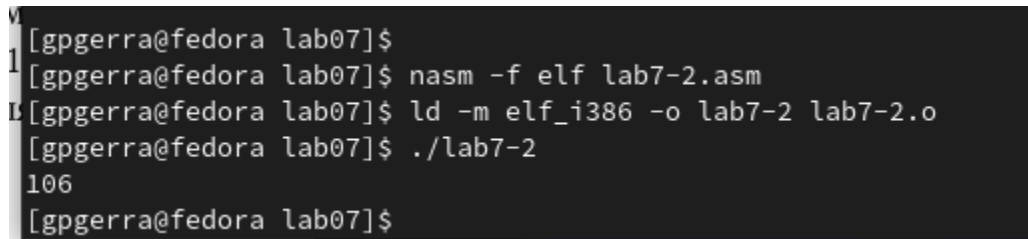
SECTION .text
GLOBAL _start
_start:

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

call quit
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9МенюМС10Выход

Рис. 8: 4.8

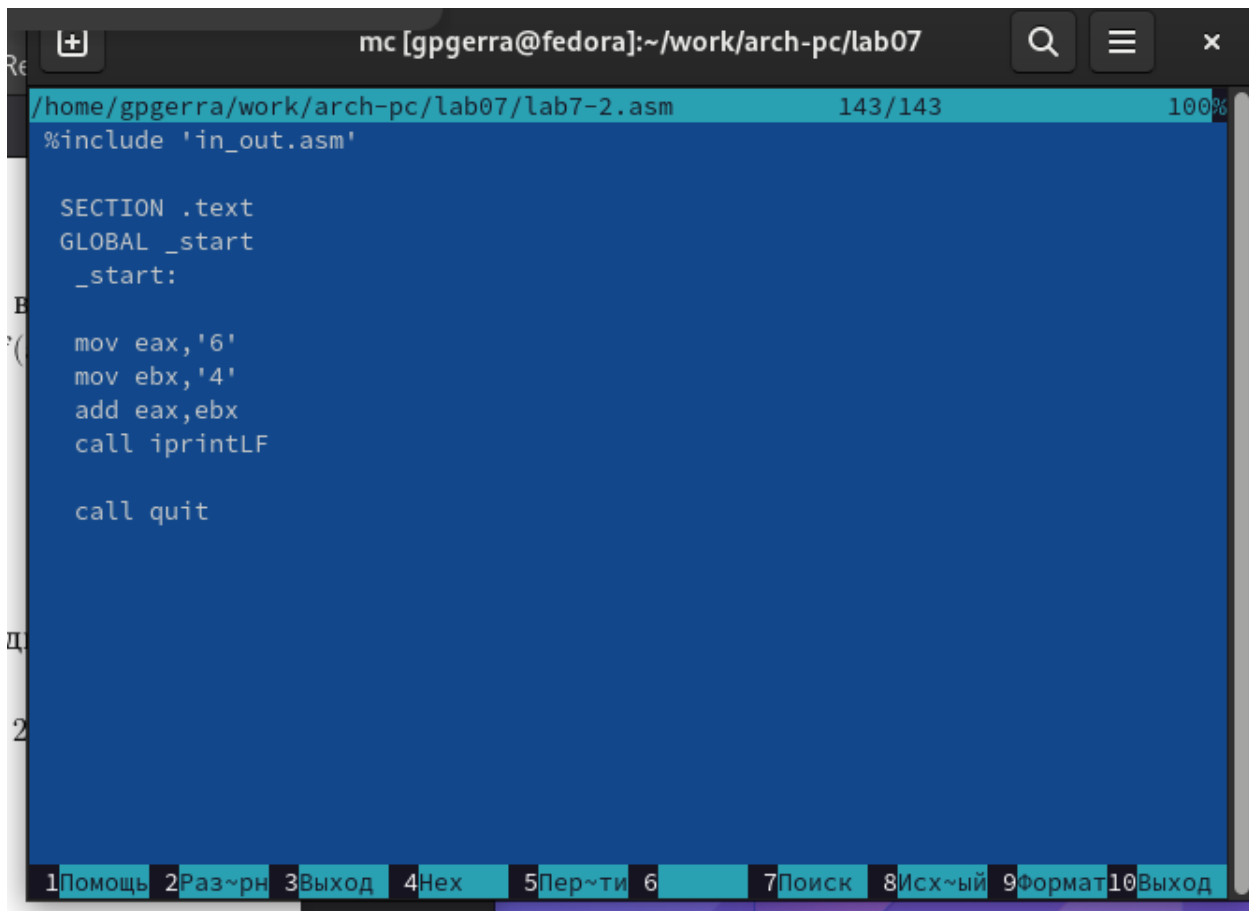


```
[gpgerra@fedora lab07]$
1 [gpgerra@fedora lab07]$ nasm -f elf lab7-2.asm
2 [gpgerra@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
3 [gpgerra@fedora lab07]$ ./lab7-2
106
[gpgerra@fedora lab07]$
```

Рис. 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. 10) (рис. 11)



The screenshot shows a text editor window titled "mc [gpgerra@fedora]:~/work/arch-pc/lab07". The file being edited is "/home/gpgerra/work/arch-pc/lab07/lab7-2.asm" (143/143 lines, 100% zoom). The code content is as follows:

```
%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 window, there is a menu bar with the following items: 1Помощь, 2Раз~рн, 3Выход, 4Нех, 5Пер~ти, 6, 7Поиск, 8Исх~ый, 9Формат, 10Выход.

Рис. 10: 4.10

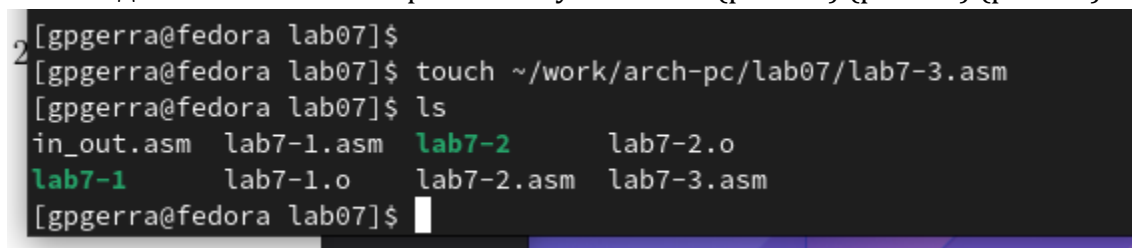
Рис. 11: 4.11

5. Заменяла функцию `iprintLF` на `iprint`. Создала исполняемый файл и запустила его. Выяснила чем отличается вывод функций `iprintLF` и `iprint` (рис. 12) (рис. 13)

Рис. 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. 14) (рис. 15) (рис. 16)



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

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

Рис. 14: 4.14


```
gerra@fedora]:~/work/arch-pc/lab07
; 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:

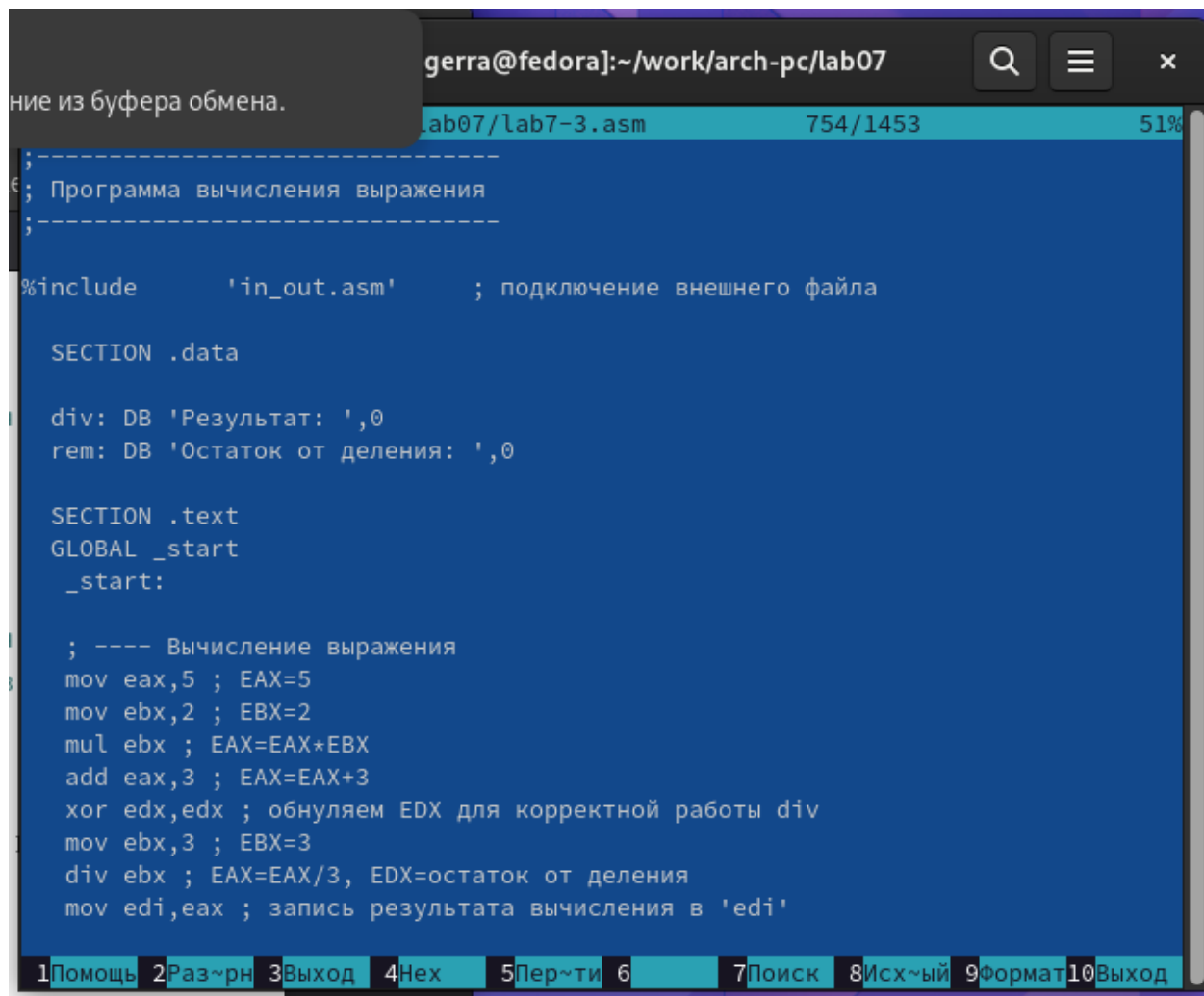
; ---- Вычисление выражения
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
```

Рис. 15: 4.15

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

Рис. 16: 4.16

7. Изменила файл так, чтобы программа вычисляла выражение $\frac{2(2)}{5} = (4 * 186 + 2)/5$ (рис. 17) (рис. 18)



```
gerra@fedora]:~/work/arch-pc/lab07
lab07/lab7-3.asm 754/1453 51%

; -----
; 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:

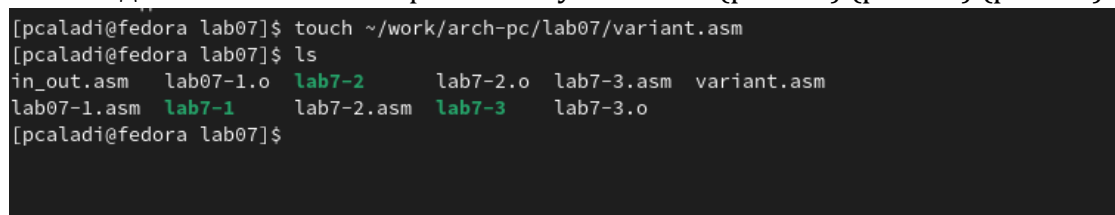
; ---- Вычисление выражения
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'

1Помощь 2Раз~рн 3Выход 4Нех 5Пер~ти 6 7Поиск 8Исх~ый 9Формат10Выход
```

Рис. 17: 4.17

Рис. 18: 4.18

8. Создала файл “вариант”, заполнила его соответственно с листингом 4, создала исполняемый файл и запустила его (рис. 19) (рис. 20) (рис. 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. 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

```

Рис. 20: 4.20

Выполнив те же вычисления вручную, выяснила, что ответ, данный програм- мой, 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 x 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 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 $5 * (x + 18) - 28$ и проверить его при $x=2$ и при $x=3$ (рис. 22) (рис. 23)

```

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. 22: 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  
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 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} :::