

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Дельгадильо Валерия

Содержание

1	Цель работы	5
2	Теоретическое введение	6
2.1	Адресация в NASM	6
2.2	Перевод символа числа в десятичную символьную запись	6
3	Лабораторной работы	8
3.1	Символьные и численные данные в NASM	8
3.2	Выполнение арифметических операций в NASM	14
4	Задание для самостоятельной работы	22
5	Выводы	26
6	Список литературы	27

Список иллюстраций

3.1	8
3.2	9
3.3	9
3.4	10
3.5	10
3.6	11
3.7	11
3.8	12
3.9	12
3.10	13
3.11	13
3.12	14
3.13	14
3.14	15
3.15	15
3.16	16
3.17	17
3.18	18
3.19	18
3.20	19
3.21	19
3.22	20
4.1	22
4.2	23
4.3	24
4.4	24

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Теоретическое введение

2.1 Адресация в NASM

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. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации.

Существует три основных способа адресации:

- 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.

2.2 Перевод символа числа в десятичную символьную запись

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производит-

ся 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.

Расширенная таблица ASCII состоит из двух частей. Первая (символы с кодами 0-127) является универсальной (см. Приложение.), а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться.

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.

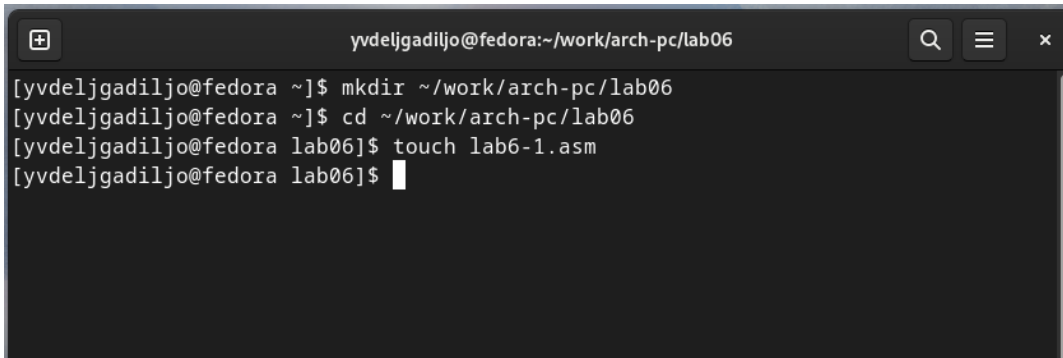
Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций.

Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.).

3 Лабораторной работы

3.1 Символьные и численные данные в NASM

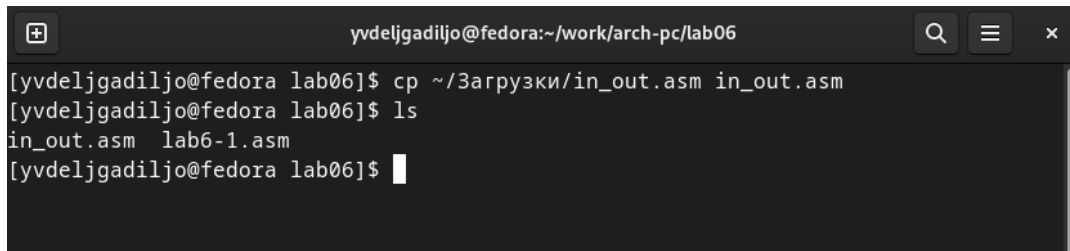
С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6. Перехожу в созданный каталог с помощью утилиты `cd`. С помощью утилиты `touch` создаю файл `lab6-1.asm`.

A screenshot of a terminal window with a dark background. The window title is "yvdeljgatiljo@fedora:~/work/arch-pc/lab06". The terminal shows the following commands and their outputs:

```
[yvdeljgatiljo@fedora ~]$ mkdir ~/work/arch-pc/lab06  
[yvdeljgatiljo@fedora ~]$ cd ~/work/arch-pc/lab06  
[yvdeljgatiljo@fedora lab06]$ touch lab6-1.asm  
[yvdeljgatiljo@fedora lab06]$
```

Рис. 3.1:

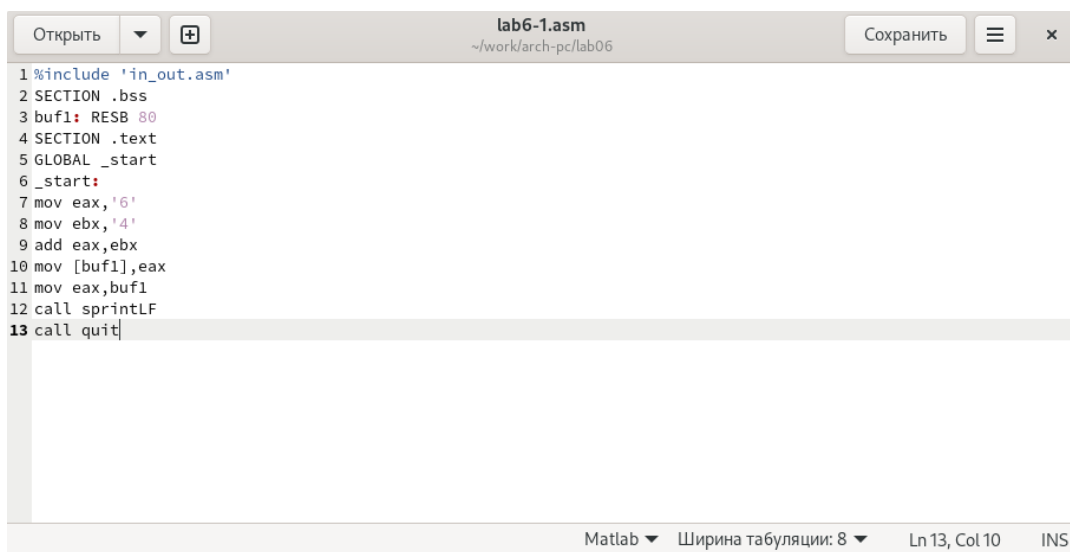
Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться в других программах.



```
yvdeljgatiljo@fedora:~/work/arch-pc/lab06
[yvdeljgatiljo@fedora lab06]$ cp ~/Загрузки/in_out.asm in_out.asm
[yvdeljgatiljo@fedora lab06]$ ls
in_out.asm  lab6-1.asm
[yvdeljgatiljo@fedora lab06]$
```

Рис. 3.2:

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



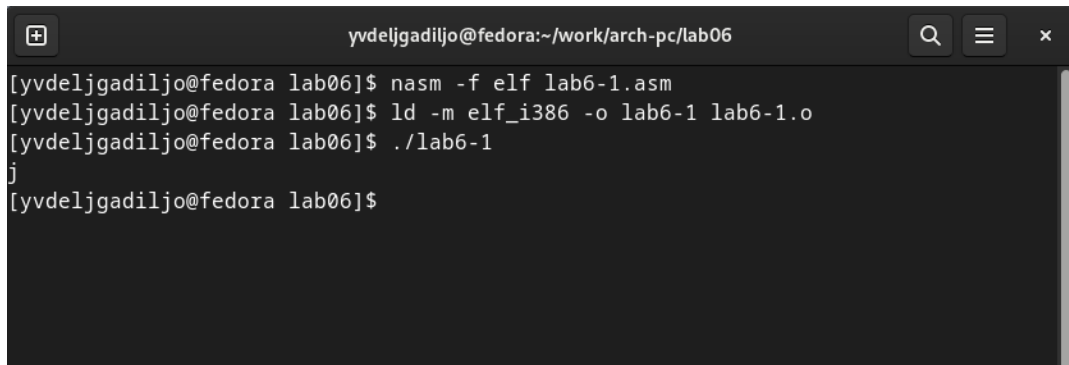
```
lab6-1.asm
~/work/arch-pc/lab06

1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintf
13 call quit

Matlab  Ширина табуляции: 8  Ln 13, Col 10  INS
```

Рис. 3.3:

Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.



```
yvdeljgadiljo@fedora:~/work/arch-pc/lab06
[yvdeljgadiljo@fedora lab06]$ nasm -f elf lab6-1.asm
[yvdeljgadiljo@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[yvdeljgadiljo@fedora lab06]$ ./lab6-1
j
[yvdeljgadiljo@fedora lab06]$
```

Рис. 3.4:

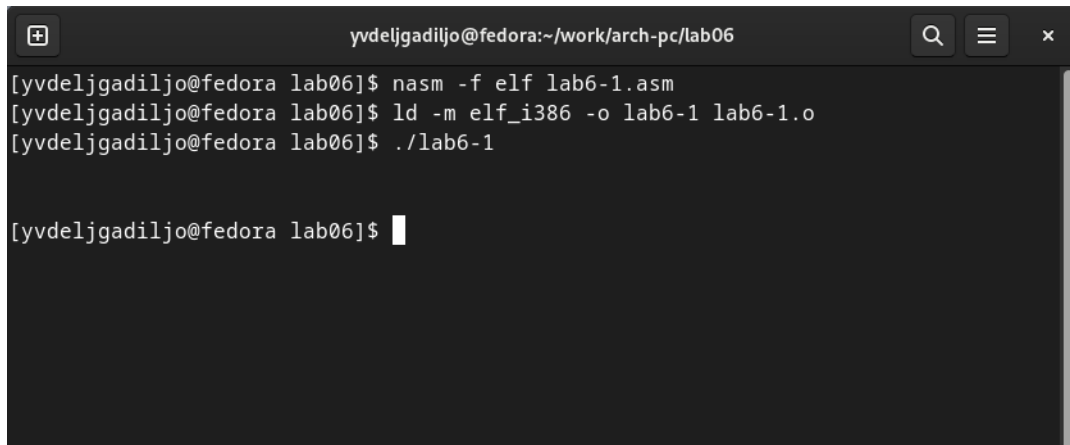
Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4.



```
*lab6-1.asm
~/work/arch-pc/lab06
Открыть Сохранить
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 mov [buf1],eax
11 mov eax,buf1
12 call sprintLF
13 call quit
```

Рис. 3.5:

Создаю новый исполняемый файл программы и запускаю его. Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

A terminal window with a dark background. The title bar shows the user 'yvdeljgadiljo' on a 'fedora' machine in the directory '~/work/arch-pc/lab06'. The terminal contains the following commands and their outputs:

```
[yvdeljgadiljo@fedora lab06]$ nasm -f elf lab6-1.asm
[yvdeljgadiljo@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[yvdeljgadiljo@fedora lab06]$ ./lab6-1

[yvdeljgadiljo@fedora lab06]$
```

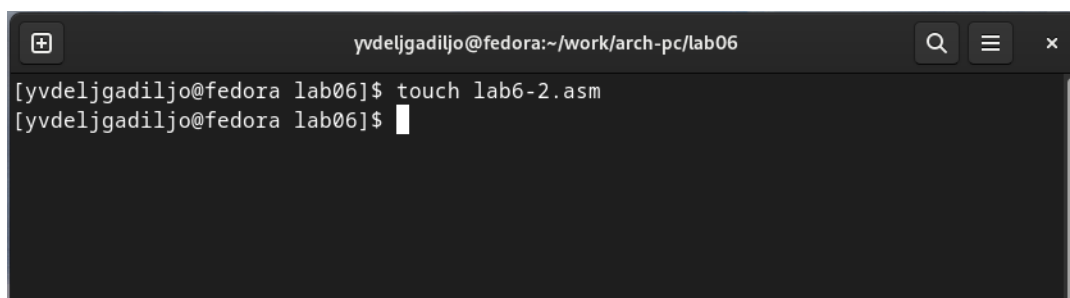
Рис. 3.6:

Создайте исполняемый файл и запустите его.

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Используя таблицу ASCII, коду 10 соответствует символ “LF” (новая строка).

Как отмечалось выше, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 6.1 с использованием этих функций.

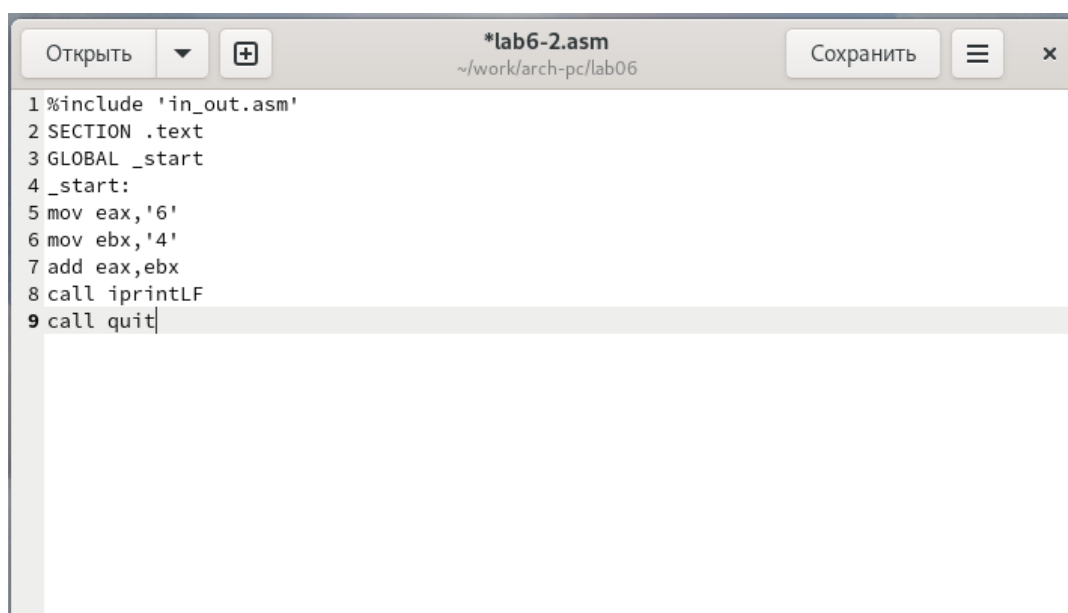
Создаю новый файл `lab6-2.asm` с помощью утилиты `touch`.

A terminal window with a dark background. The title bar shows the user 'yvdeljgadiljo' on a 'fedora' machine in the directory '~/work/arch-pc/lab06'. The terminal contains the following commands and their outputs:

```
[yvdeljgadiljo@fedora lab06]$ touch lab6-2.asm
[yvdeljgadiljo@fedora lab06]$
```

Рис. 3.7:

Ввожу в файл текст другой программы для вывода значения регистра `eax`.

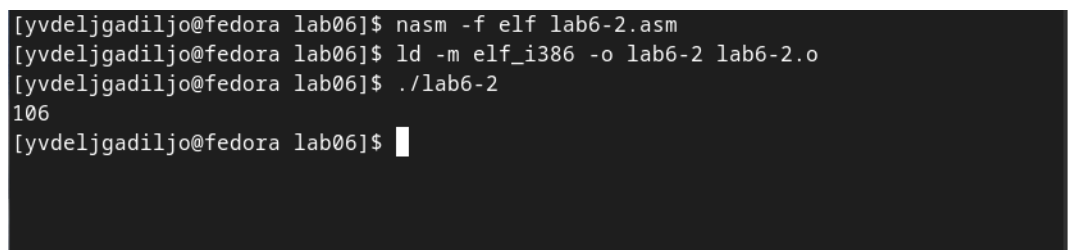


The screenshot shows a text editor window titled '*lab6-2.asm' with a path of '~/.work/arch-pc/lab06'. The editor contains the following assembly code:

```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, '6'
6 mov ebx, '4'
7 add eax, ebx
8 call iprintLF
9 call quit
```

Рис. 3.8:

Создаю и запускаю исполняемый файл lab6-2. 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”.



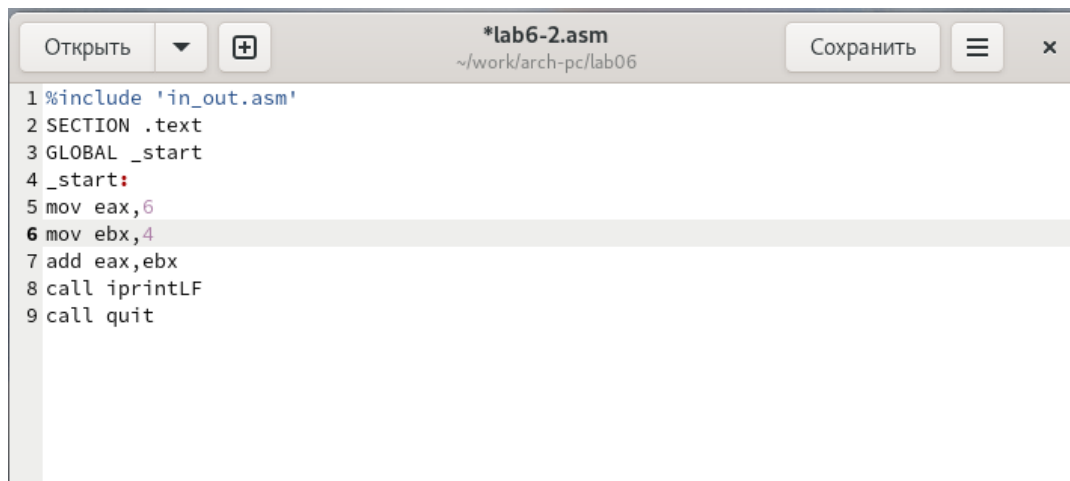
The screenshot shows a terminal window with the following commands and output:

```
[yvdeljgatiljo@fedora lab06]$ nasm -f elf lab6-2.asm
[yvdeljgatiljo@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[yvdeljgatiljo@fedora lab06]$ ./lab6-2
106
[yvdeljgatiljo@fedora lab06]$
```

Рис. 3.9:

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда `add` складывает коды символов ‘6’ и ‘4’ ($54+52=106$). Однако, в отличии от программы из листинга 6.1, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

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

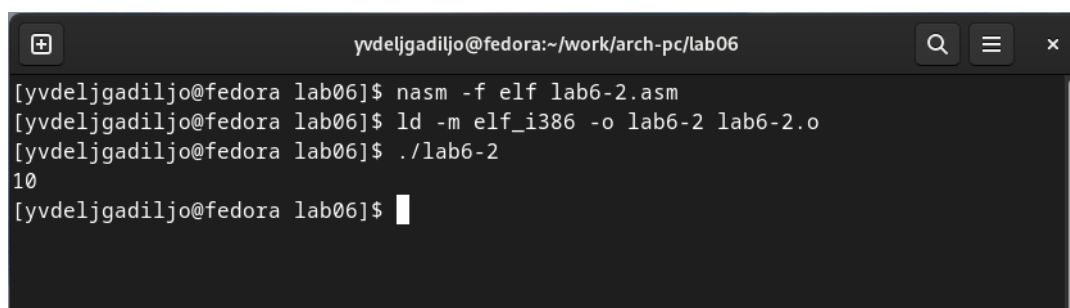


The screenshot shows a text editor window titled '*lab6-2.asm' with the path '~/work/arch-pc/lab06'. The code is as follows:

```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprintLF
9 call quit
```

Рис. 3.10:

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



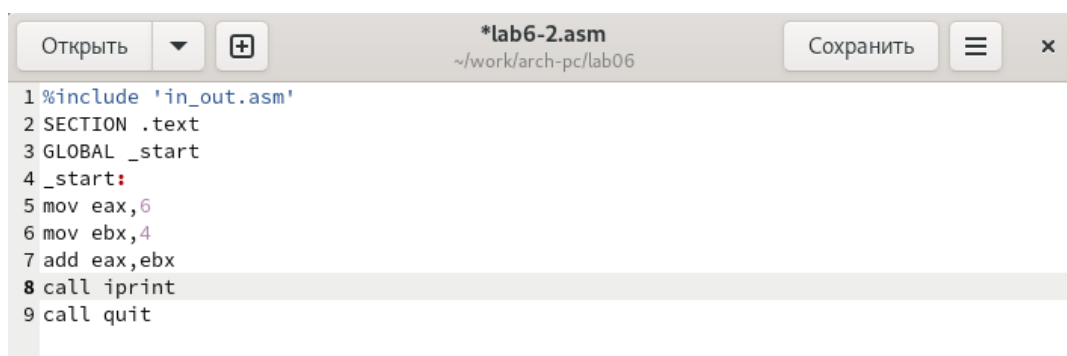
The screenshot shows a terminal window with the following commands and output:

```
yvdeljgatiljo@fedora:~/work/arch-pc/lab06
[yvdeljgatiljo@fedora lab06]$ nasm -f elf lab6-2.asm
[yvdeljgatiljo@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[yvdeljgatiljo@fedora lab06]$ ./lab6-2
10
[yvdeljgatiljo@fedora lab06]$
```

Рис. 3.11:

Создайте исполняемый файл и запустите его. Какой результат будет получен при исполнении программы?

Заменяю в тексте программы функцию iprintLF на iprint.

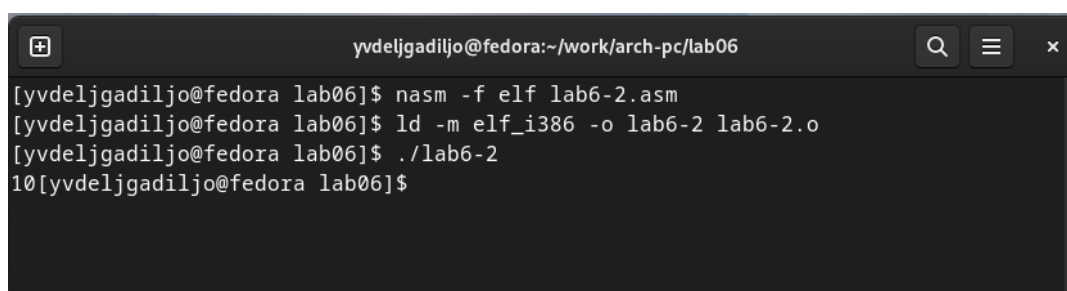


```
*lab6-2.asm
~/work/arch-pc/lab06

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprint
9 call quit
```

Рис. 3.12:

Создаю и запускаю новый исполняемый файл. 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`.



```
yvdeljgadiljo@fedora:~/work/arch-pc/lab06

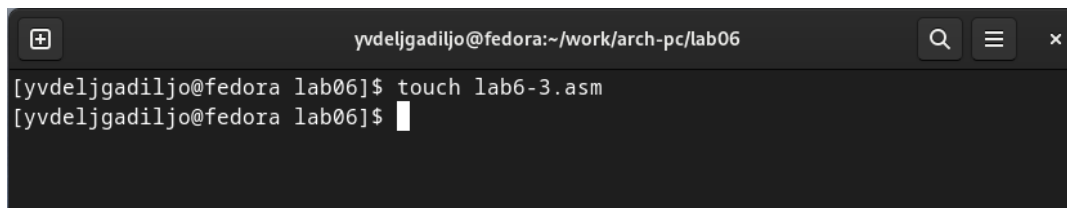
[yvdeljgadiljo@fedora lab06]$ nasm -f elf lab6-2.asm
[yvdeljgadiljo@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[yvdeljgadiljo@fedora lab06]$ ./lab6-2
10[yvdeljgadiljo@fedora lab06]$
```

Рис. 3.13:

3.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5*2 + 3)/3$.

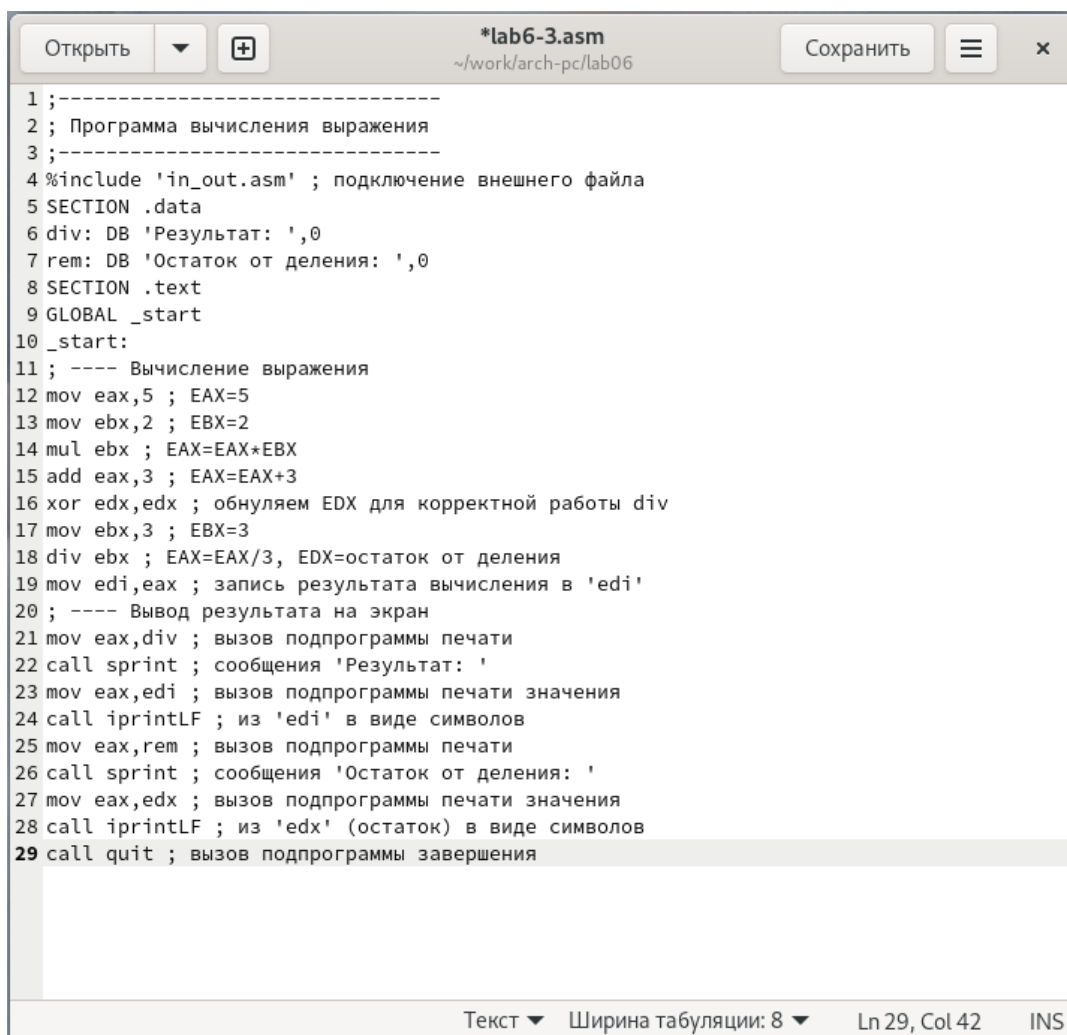
Создаю файл `lab7-3.asm` с помощью утилиты `touch`.



```
yvdeljgatiljo@fedora:~/work/arch-pc/lab06
[yvdeljgatiljo@fedora lab06]$ touch lab6-3.asm
[yvdeljgatiljo@fedora lab06]$
```

Рис. 3.14:

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

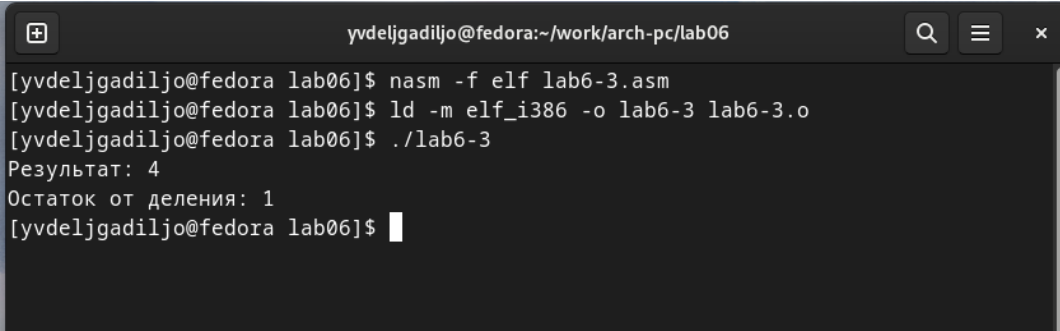


```
*lab6-3.asm
~/work/arch-pc/lab06

1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4 %include 'in_out.asm' ; подключение внешнего файла
5 SECTION .data
6 div: DB 'Результат: ',0
7 rem: DB 'Остаток от деления: ',0
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ; ---- Вычисление выражения
12 mov eax,5 ; EAX=5
13 mov ebx,2 ; EBX=2
14 mul ebx ; EAX=EAX*EBX
15 add eax,3 ; EAX=EAX+3
16 xor edx,edx ; обнуляем EDX для корректной работы div
17 mov ebx,3 ; EBX=3
18 div ebx ; EAX=EAX/3, EDX=остаток от деления
19 mov edi,eax ; запись результата вычисления в 'edi'
20 ; ---- Вывод результата на экран
21 mov eax,div ; вызов подпрограммы печати
22 call sprint ; сообщения 'Результат: '
23 mov eax,edi ; вызов подпрограммы печати значения
24 call iprintLF ; из 'edi' в виде символов
25 mov eax,rem ; вызов подпрограммы печати
26 call sprint ; сообщения 'Остаток от деления: '
27 mov eax,edx ; вызов подпрограммы печати значения
28 call iprintLF ; из 'edx' (остаток) в виде символов
29 call quit ; вызов подпрограммы завершения
```

Рис. 3.15:

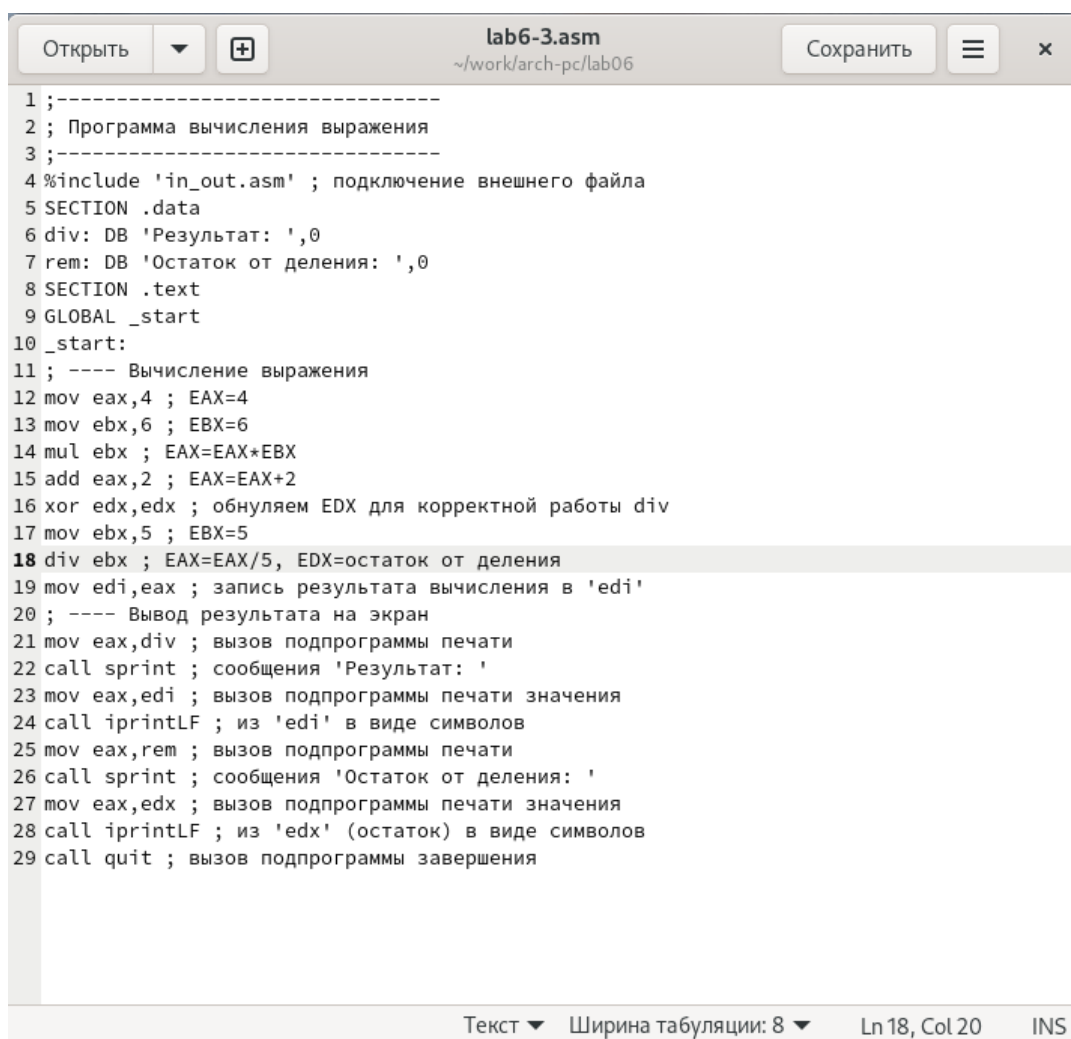
Создаю исполняемый файл и запускаю его.



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

Рис. 3.16:

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

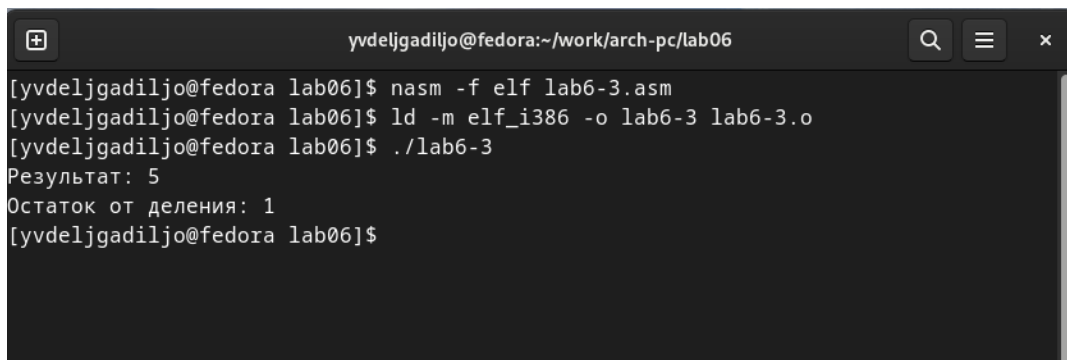


```
1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4 %include 'in_out.asm' ; подключение внешнего файла
5 SECTION .data
6 div: DB 'Результат: ',0
7 rem: DB 'Остаток от деления: ',0
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ; ---- Вычисление выражения
12 mov eax,4 ; EAX=4
13 mov ebx,6 ; EBX=6
14 mul ebx ; EAX=EAX*EBX
15 add eax,2 ; EAX=EAX+2
16 xor edx,edx ; обнуляем EDX для корректной работы div
17 mov ebx,5 ; EBX=5
18 div ebx ; EAX=EAX/5, EDX=остаток от деления
19 mov edi,eax ; запись результата вычисления в 'edi'
20 ; ---- Вывод результата на экран
21 mov eax,div ; вызов подпрограммы печати
22 call sprint ; сообщения 'Результат: '
23 mov eax,edi ; вызов подпрограммы печати значения
24 call iprintLF ; из 'edi' в виде символов
25 mov eax,rem ; вызов подпрограммы печати
26 call sprint ; сообщения 'Остаток от деления: '
27 mov eax,edx ; вызов подпрограммы печати значения
28 call iprintLF ; из 'edx' (остаток) в виде символов
29 call quit ; вызов подпрограммы завершения
```

Текст ▾ Ширина табуляции: 8 ▾ Ln 18, Col 20 INS

Рис. 3.17:

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

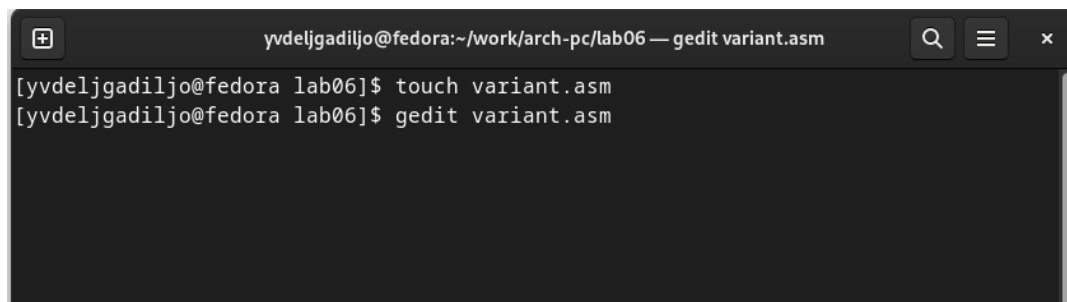
A terminal window with a dark background. The title bar shows the user 'yvdeljgadiljo' on a 'fedora' machine in the directory '~/work/arch-pc/lab06'. The terminal contains the following text:

```
[yvdeljgadiljo@fedora lab06]$ nasm -f elf lab6-3.asm
[yvdeljgadiljo@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[yvdeljgadiljo@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
[yvdeljgadiljo@fedora lab06]$
```

Рис. 3.18:

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция `atoi` из файла `in_out.asm`.

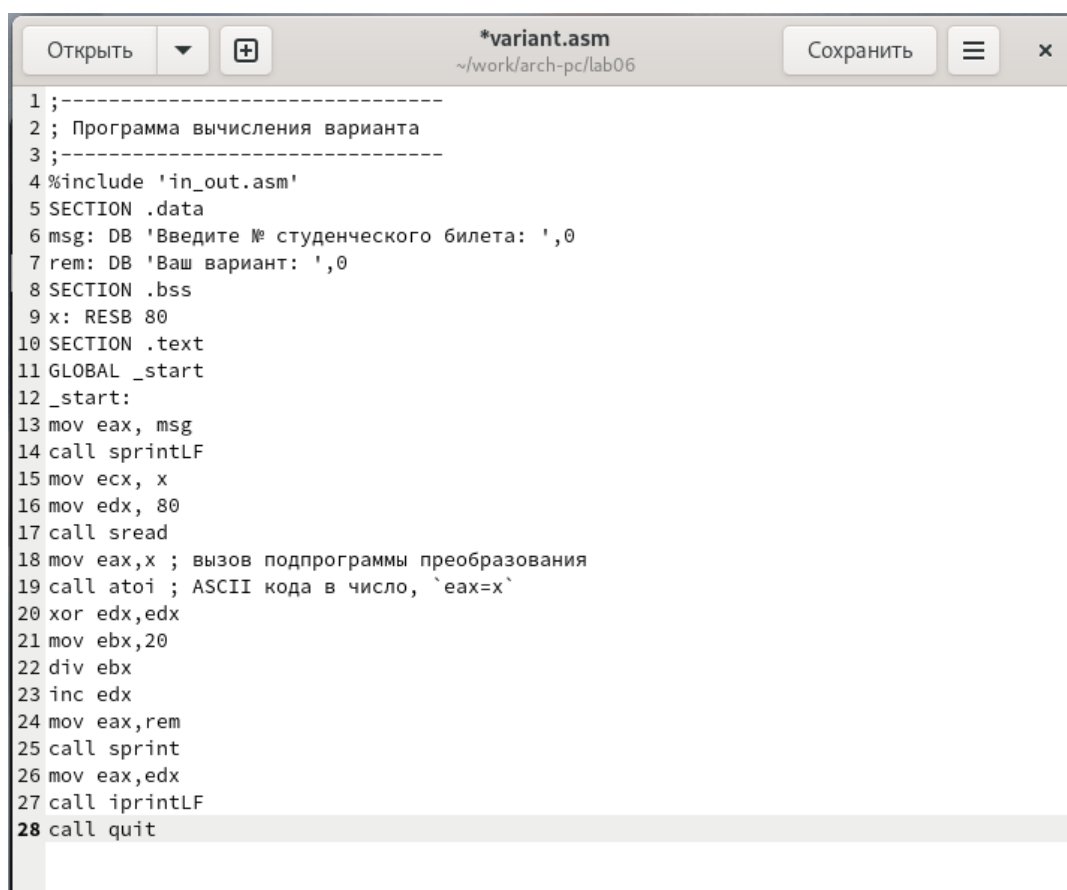
Создаю файл `variant.asm` с помощью утилиты `touch`.

A terminal window with a dark background. The title bar shows the user 'yvdeljgadiljo' on a 'fedora' machine in the directory '~/work/arch-pc/lab06', and it also indicates that the file 'variant.asm' is being edited with 'gedit'. The terminal contains the following text:

```
[yvdeljgadiljo@fedora lab06]$ touch variant.asm
[yvdeljgadiljo@fedora lab06]$ gedit variant.asm
```

Рис. 3.19:

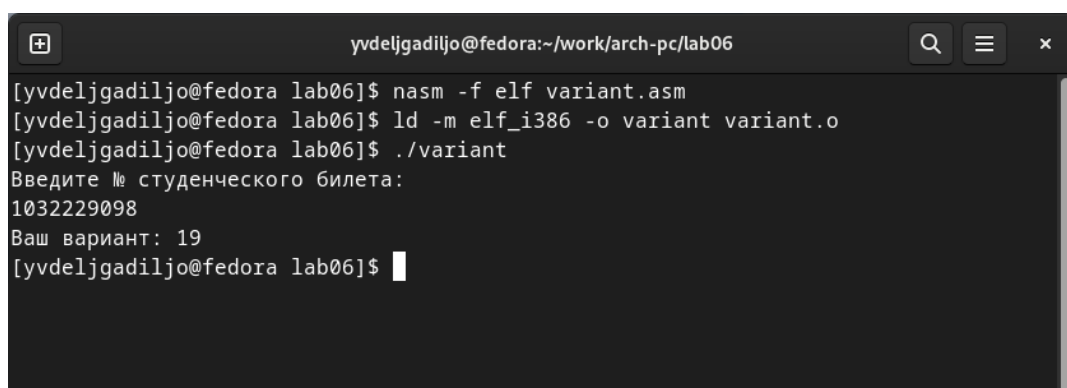
Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета.



```
1 ;-----
2 ; Программа вычисления варианта
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg: DB 'Введите № студенческого билета: ',0
7 rem: DB 'Ваш вариант: ',0
8 SECTION .bss
9 x: RESB 80
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprintLF
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax, x ; вызов подпрограммы преобразования
19 call atoi ; ASCII кода в число, `eax=x`
20 xor edx, edx
21 mov ebx, 20
22 div ebx
23 inc edx
24 mov eax, rem
25 call sprint
26 mov eax, edx
27 call iprintLF
28 call quit
```

Рис. 3.20:

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 19.



```
yvdeljgatiljo@fedora:~/work/arch-pc/lab06
[yvdeljgatiljo@fedora lab06]$ nasm -f elf variant.asm
[yvdeljgatiljo@fedora lab06]$ ld -m elf_i386 -o variant variant.o
[yvdeljgatiljo@fedora lab06]$ ./variant
Введите № студенческого билета:
1032229098
Ваш вариант: 19
[yvdeljgatiljo@fedora lab06]$
```

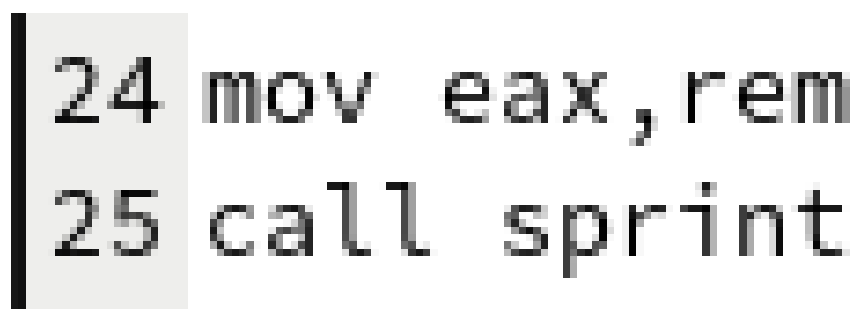
Рис. 3.21:

Создайте исполняемый файл и запустите его. Проверьте результат работы программы вычислив номер варианта аналитически.

Включите в отчет по выполнению лабораторной работы ответы на следующие вопросы:

- Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

За вывод сообщения “Ваш вариант” отвечают строки кода:



```
24 mov eax, rem
25 call sprint
```

Рис. 3.22:

- Для чего используются следующие инструкции?

```
mov ecx, x
```

```
mov edx, 80
```

```
call sread
```

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.

- Для чего используется инструкция “`call atoi`”?

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

- Какие строки листинга 6.4 отвечают за вычисления варианта?

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

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

При выполнении инструкции div ebx остаток от деления записывается в регистр edx

- Для чего используется инструкция “inc edx”?

Инструкция inc edx увеличивает значение регистра edx на 1

- Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

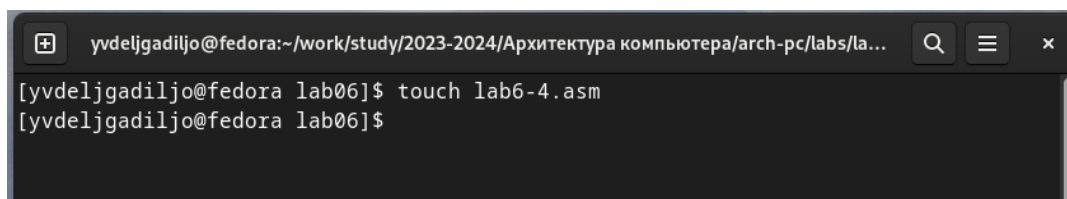
За вывод на экран результатов вычислений отвечают строки:

mov eax,edx

call iprintLF

4 Задание для самостоятельной работы

Написать программу вычисления выражения $y = f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создаю файл lab7-4.asm с помощью утилиты touch.



```
yvdeljgatiljo@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/la...
[yvdeljgatiljo@fedora lab06]$ touch lab6-4.asm
[yvdeljgatiljo@fedora lab06]$
```

Рис. 4.1:

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $((1/3)x + 5) * 7$ Это выражение было под вариантом 19.

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg: DB 'Введите значение переменной x: ',0
4 rem: DB 'Результат: ',0
5 SECTION .bss
6 x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры
7 SECTION .text
8 GLOBAL _start
9 _start:
10 ; ---- Вычисление выражения
11 mov eax, msg
12 call sprint
13 mov ecx, x
14 mov edx, 80
15 call sread
16 mov eax, x ; вызов подпрограммы преобразования
17 call atoi ; ASCII кода в число, 'eax=x'
18 xor edx,edx ; обнуляем EDX для корректной работы div
19 mov ebx,3 ; EBX=3
20 div ebx ; EAX=EAX/3, EDX=остаток от деления
21 add eax,5 ; EAX=EAX+5 = (x/3)+5
22 mov ebx,7 ; EBX=7
23 mul ebx ; EAX=EAX*EBX = ((x/3)+5)*7
24 mov edi,eax ; запись результата вычисления в 'edi'
25 ; ---- Вывод результата на экран
26 mov eax,rem ; вызов подпрограммы печати
27 call sprint ; сообщения 'Результат: '
28 mov eax,edi ; вызов подпрограммы печати значения
29 call iprintLF ; из 'edi' в виде символов
30 call quit ; вызов подпрограммы завершения
31
```

Matlab ▾ Ширина табуляции: 8 ▾ Ln 29, Col 14 INS

Рис. 4.2:

Создаю и запускаю исполняемый файл. При вводе значения 3, вывод - 42.

```
yvdeljgatiljo@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/la...
[yvdeljgatiljo@fedora lab06]$ nasm -f elf lab6-4.asm
[yvdeljgatiljo@fedora lab06]$ ld -m elf_i386 -o lab6-4 lab6-4.o
[yvdeljgatiljo@fedora lab06]$ ./lab6-4
Введите значение переменной x: 3
Результат: 42
[yvdeljgatiljo@fedora lab06]$
```

Рис. 4.3:

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

```
[yvdeljgatiljo@fedora lab06]$ nasm -f elf lab6-4.asm
[yvdeljgatiljo@fedora lab06]$ ld -m elf_i386 -o lab6-4 lab6-4.o
[yvdeljgatiljo@fedora lab06]$ ./lab6-4
Введите значение переменной x: 9
Результат: 56
[yvdeljgatiljo@fedora lab06]$
```

Рис. 4.4:

****Листинг 4.1. Программа для вычисления значения выражения $((x/3)+5)^7$.****

%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 ; Переменная, значение которой будем вводить с клавиатуры

SECTION .text

GLOBAL _start

_start:


```

; ---- Вычисление выражения
mov eax, msg
call sprint
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 ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
add eax,5 ; EAX=EAX+5 = (x/3)+5
mov ebx,7 ; EBX=7
mul ebx ; EAX=EAX*EBX = ((x/3)+5)*7
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 iprintLF ; 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

- GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
- GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
- Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
- NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
- Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
- Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
- The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
- Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
- Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
- Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
- Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
- Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.

- Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
- Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
- Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
- Таненбаум Э., Бос Х. Современные операционн