

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Курушин Георгий Романович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Символьные и численные данные в NASM	9
4.2	Выполнение арифметических операций в NASM	14
4.3	Ответы на контрольные вопросы	18
4.4	Задание для самостоятельной работы	19
5	Выводы	22
6	Список литературы	23

Список иллюстраций

4.1	Создание нового каталога	9
4.2	Сохранение новой программы	10
4.3	Запуск изначальной программы	10
4.4	Измененная программа	11
4.5	Запуск измененной программы	11
4.6	Вторая программа	12
4.7	Вывод второй программы	13
4.8	Вывод измененной второй программы	13
4.9	Замена функции вывода во второй программе	14
4.10	Третья программа	15
4.11	Запуск третьей программы	15
4.12	Изменение третьей программы	16
4.13	Запуск измененной третьей программы	16
4.14	Программа для подсчета варианта	17
4.15	Запуск программы для подсчета варианта	18
4.16	Запуск и проверка программы	20

Список таблиц

1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

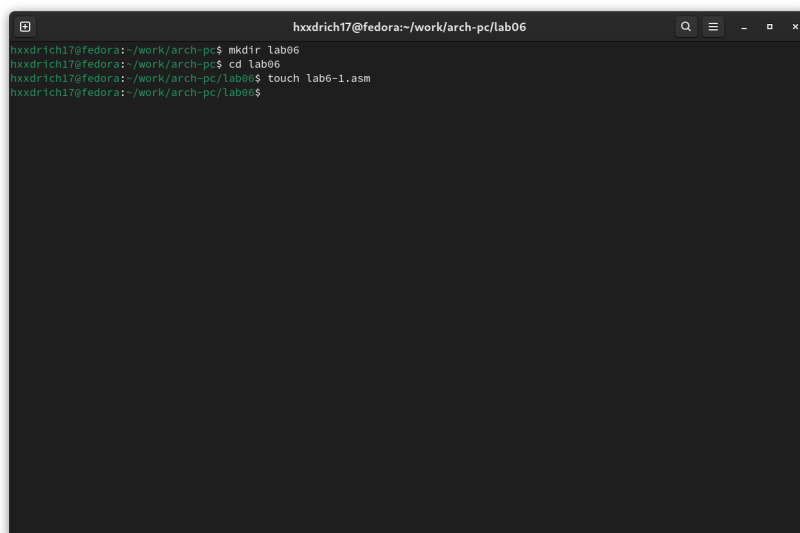
Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена 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. Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится 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. 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.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

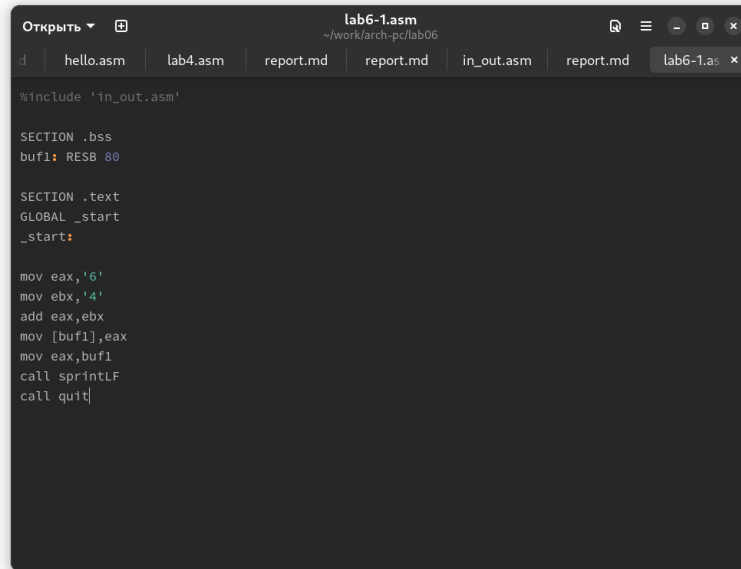
Создаю каталог для программ лабораторной работы №6 и перехожу в него, создаю там файл (рис. -fig. 4.1).

A screenshot of a terminal window with a dark background. The window title is 'hxxdrich17@fedora:~/work/arch-pc/lab06'. The terminal shows the following commands and their outputs: 'mkdir lab06', 'cd lab06', and 'touch lab6-1.asm'. The prompt changes from '~/' to '~/work/arch-pc/lab06' after the 'cd' command.

```
hxxdrich17@fedora:~/work/arch-pc/lab06
hxxdrich17@fedora:~/work/arch-pc$ mkdir lab06
hxxdrich17@fedora:~/work/arch-pc$ cd lab06
hxxdrich17@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
hxxdrich17@fedora:~/work/arch-pc/lab06$
```

Рис. 4.1: Создание нового каталога

В созданном файле ввожу программу из листинга (рис. -fig. 4.2).



```
Открыть ▾ lab6-1.asm ~/work/arch-pc/lab06
hello.asm lab4.asm report.md report.md in_out.asm report.md lab6-1.as x
%include 'in_out.asm'

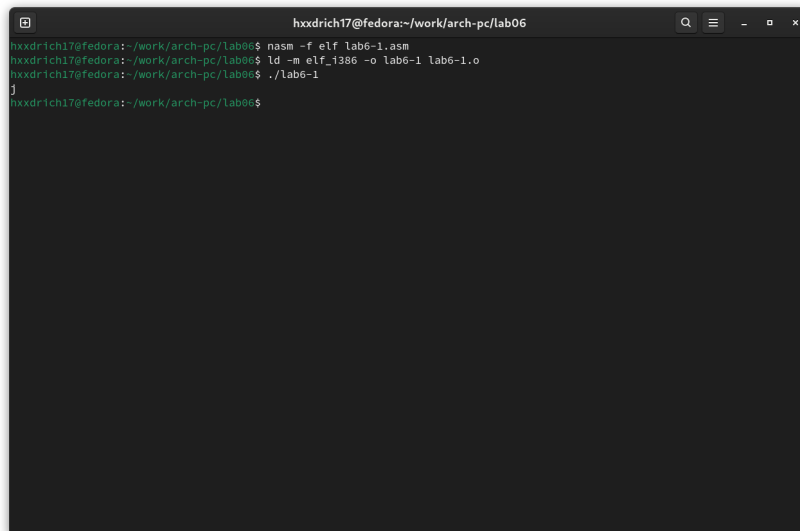
SECTION .bss
buf1: RESB 80

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. 4.2: Сохранение новой программы

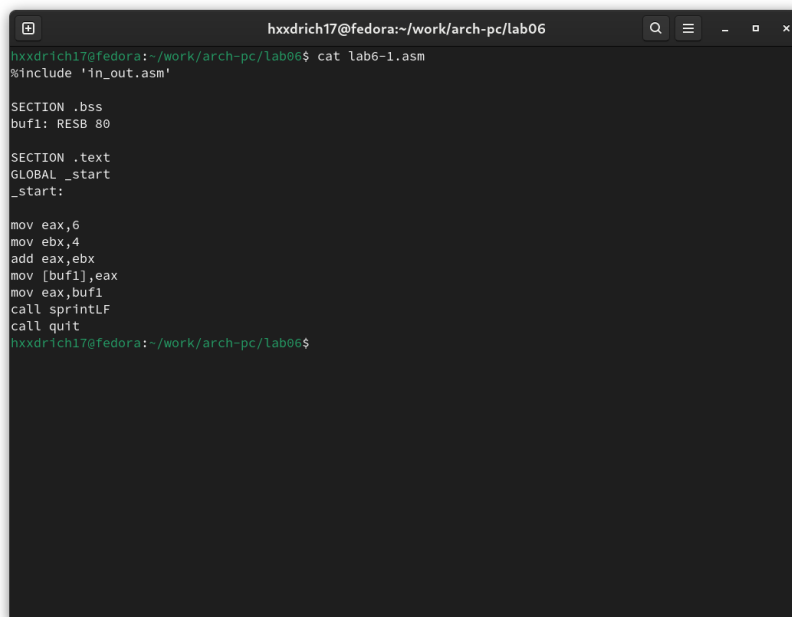
Создаю исполняемый файл и запускаю его, вывод программы отличается от предполагаемого изначально, ибо коды символов в сумме дают символ j по таблице ASCII. {#fig:003 width=70%}



```
hxxdrich17@fedora:~/work/arch-pc/lab06
hxxdrich17@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
hxxdrich17@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
hxxdrich17@fedora:~/work/arch-pc/lab06$ ./lab6-1
j
hxxdrich17@fedora:~/work/arch-pc/lab06$
```

Рис. 4.3: Запуск изначальной программы

Изменяю текст изначальной программы, убрав кавычки (рис. -fig. 4.4).

A terminal window with a dark background and light green text. The title bar shows 'hxxdrich17@fedora:~/work/arch-pc/lab06'. The command 'cat lab6-1.asm' has been executed, displaying the following assembly code:

```
hxxdrich17@fedora:~/work/arch-pc/lab06$ cat lab6-1.asm
%include 'in_out.asm'

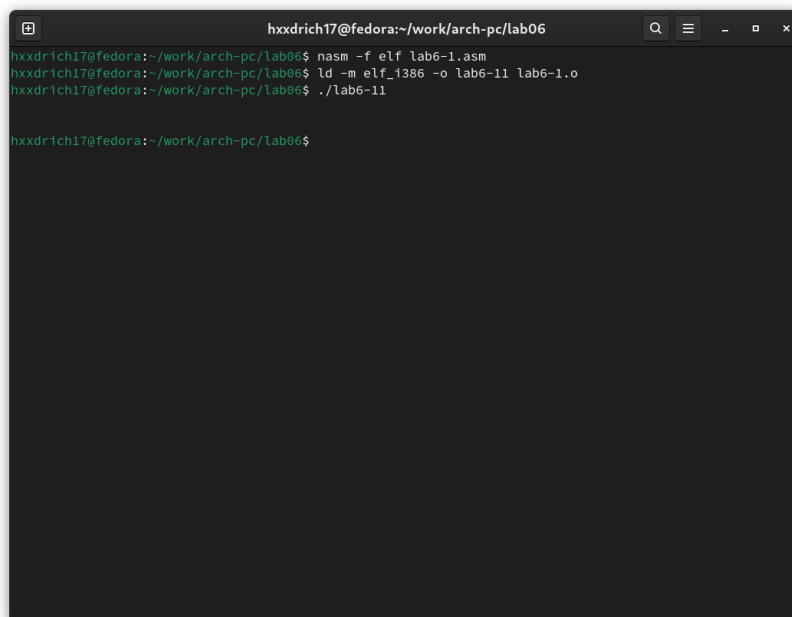
SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,6
mov ebx,4
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintf
call quit
hxxdrich17@fedora:~/work/arch-pc/lab06$
```

Рис. 4.4: Измененная программа

На этот раз программа выдала пустую строку, это связано с тем, что символ 10 означает переход на новую строку (рис. -fig. 4.5).

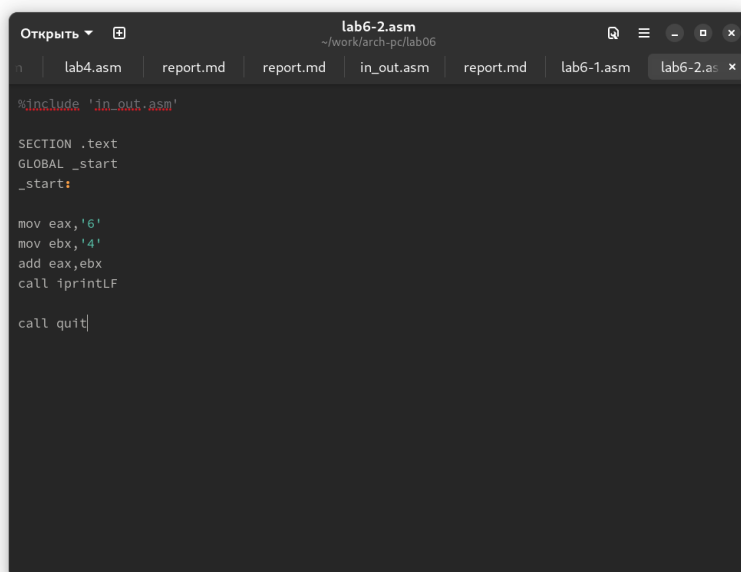
A terminal window with a dark background and light green text. The title bar shows 'hxxdrich17@fedora:~/work/arch-pc/lab06'. The following commands have been executed:

```
hxxdrich17@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
hxxdrich17@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-11 lab6-1.o
hxxdrich17@fedora:~/work/arch-pc/lab06$ ./lab6-11

hxxdrich17@fedora:~/work/arch-pc/lab06$
```

Рис. 4.5: Запуск измененной программы

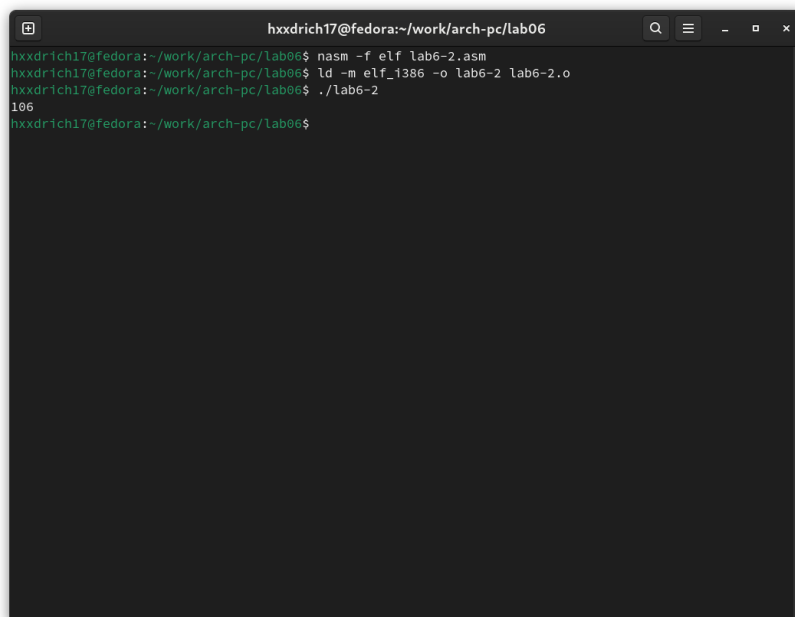
Создаю новый файл для будущей программы и записываю в нее код из листинга (рис. -fig. 4.6).



```
Открыть ▾  
lab6-2.asm  
~/work/arch-pc/lab06  
lab4.asm  report.md  report.md  in_out.asm  report.md  lab6-1.asm  lab6-2.as x  
%include 'in_out.asm'  
  
SECTION .text  
GLOBAL _start  
_start:  
  
mov eax,'6'  
mov ebx,'4'  
add eax,ebx  
call iprintLF  
  
call quit|
```

Рис. 4.6: Вторая программа

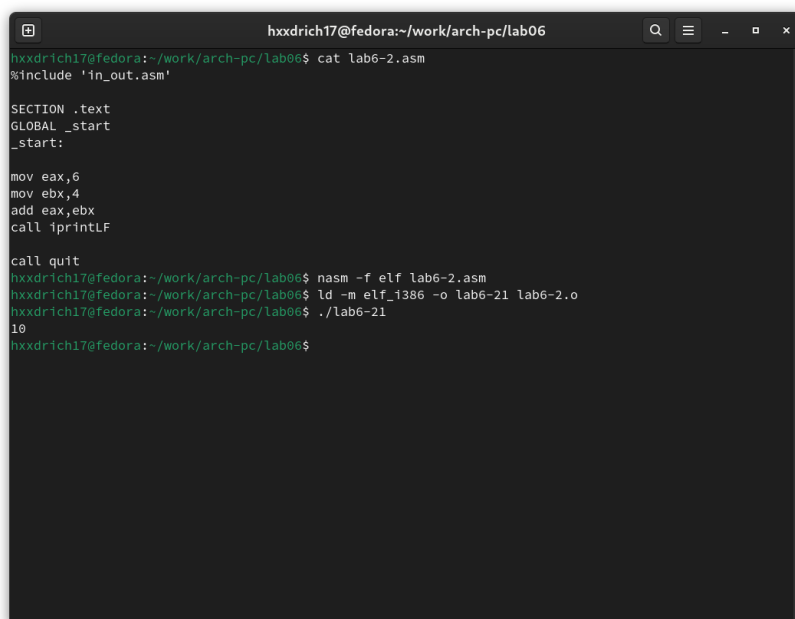
Создаю исполняемый файл и запускаю его, теперь отображается результат 106, программа, как и в первый раз, сложила коды символов, но вывела само число, а не его символ, благодаря замене функции вывода на iprintLF (рис. -fig. 4.7).

A terminal window with a dark background and light green text. The window title is 'hxxdrich17@fedora:~/work/arch-pc/lab06'. The terminal shows the following commands and their outputs:

```
hxxdrich17@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
hxxdrich17@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
hxxdrich17@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
hxxdrich17@fedora:~/work/arch-pc/lab06$
```

Рис. 4.7: Вывод второй программы

Убрав кавычки в программе, я снова ее запускаю и получаю предполагаемый изначально результат. (рис. -fig. 4.8).

A terminal window with a dark background and light green text. The window title is 'hxxdrich17@fedora:~/work/arch-pc/lab06'. The terminal shows the following commands and their outputs:

```
hxxdrich17@fedora:~/work/arch-pc/lab06$ cat lab6-2.asm
#include 'in_out.asm'

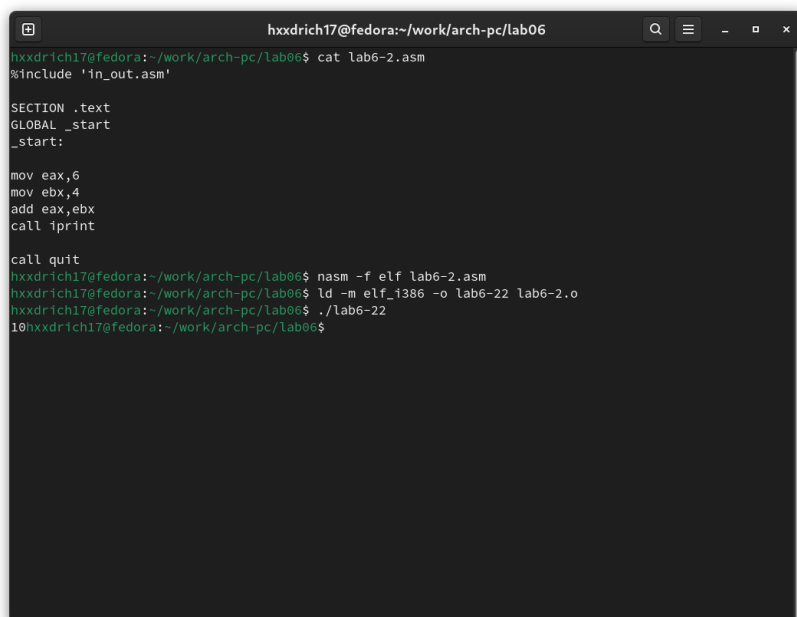
SECTION .text
GLOBAL _start
_start:

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

call quit
hxxdrich17@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
hxxdrich17@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-21 lab6-2.o
hxxdrich17@fedora:~/work/arch-pc/lab06$ ./lab6-21
10
hxxdrich17@fedora:~/work/arch-pc/lab06$
```

Рис. 4.8: Вывод измененной второй программы

Заменяв функцию вывода на `iprint`, я получаю тот же результат, но без переноса строки (рис. -fig. 4.9).



```
hxxdrich17@fedora:~/work/arch-pc/lab06
hxxdrich17@fedora:~/work/arch-pc/lab06$ cat lab6-2.asm
%include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

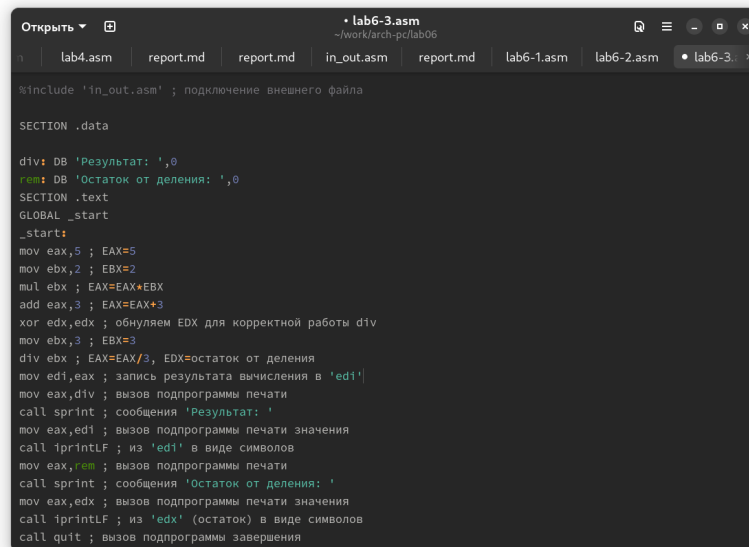
mov eax,6
mov ebx,4
add eax,ebx
call iprint

call quit
hxxdrich17@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
hxxdrich17@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-22 lab6-2.o
hxxdrich17@fedora:~/work/arch-pc/lab06$ ./lab6-22
10hxxdrich17@fedora:~/work/arch-pc/lab06$
```

Рис. 4.9: Замена функции вывода во второй программе

4.2 Выполнение арифметических операций в NASM

Создаю новый файл и копирую в него содержимое листинга (рис. -fig. 4.10).

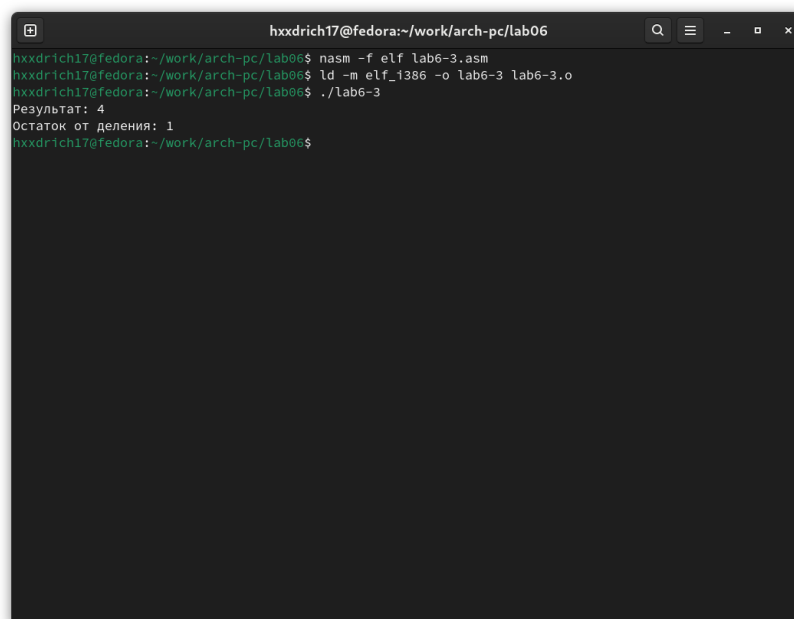


```
Открыть ▾ lab6-3.asm ~/work/arch-pc/lab06
lab4.asm report.md report.md in_out.asm report.md lab6-1.asm lab6-2.asm lab6-3.asm
#include "in_out.asm" ; подключение внешнего файла

SECTION .data
div: DB "Результат: ",0
rem: DB "Остаток от деления: ",0
SECTION .text
GLOBAL _start
_start:
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'
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения "Результат: "
mov eax,edi ; вызов подпрограммы печати значения
call iprintf ; 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 iprintf ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.10: Третья программа

Программа выполняет арифметические вычисления, на вывод идет результирующее выражения и его остаток от деления (рис. -fig. 4.11).

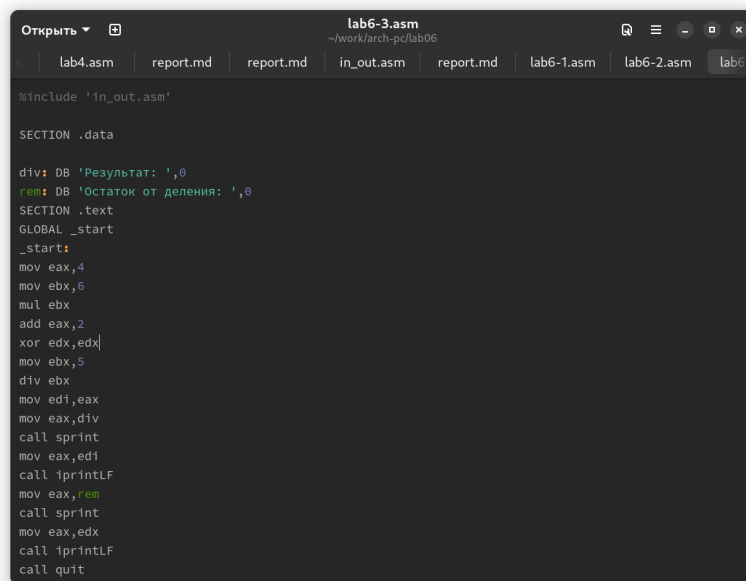


```
hxxdrich17@fedora:~/work/arch-pc/lab06
hxxdrich17@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
hxxdrich17@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
hxxdrich17@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
hxxdrich17@fedora:~/work/arch-pc/lab06$
```

Рис. 4.11: Запуск третьей программы

Заменяв переменные в программе для выражения $f(x) = (4*6+2)/5$ (рис.

-fig. 4.12).



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

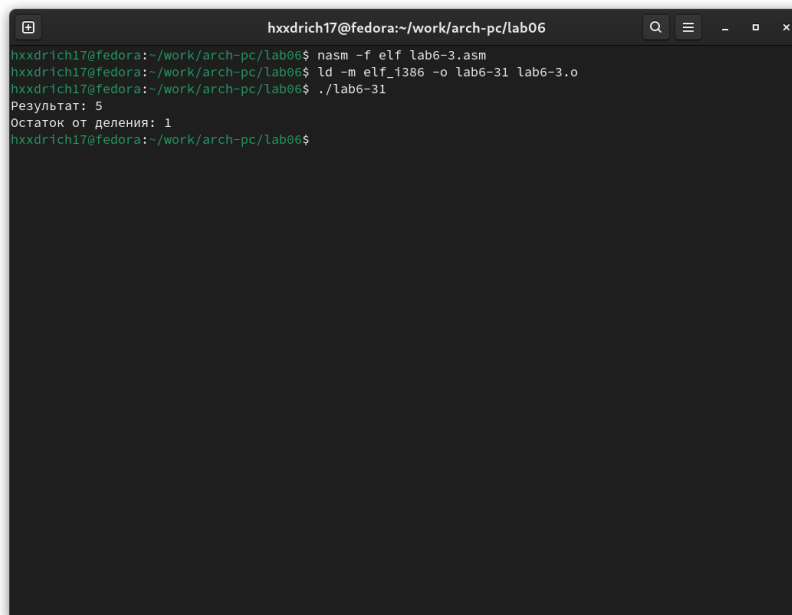
lab4.asm  report.md  report.md  in_out.asm  report.md  lab6-1.asm  lab6-2.asm  lab6-3.asm

%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Рис. 4.12: Изменение третьей программы

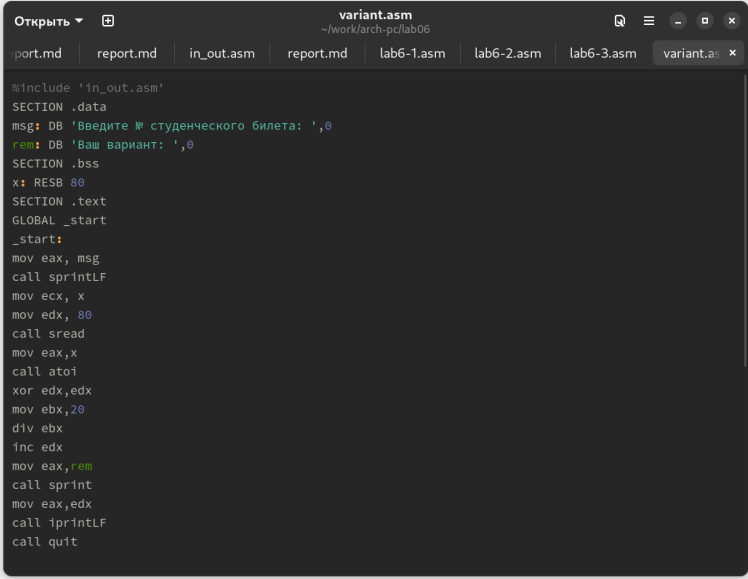
Запуск программы дает корректный результат (рис. -fig. 4.13).



```
hxxdrich17@fedora:~/work/arch-pc/lab06
hxxdrich17@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
hxxdrich17@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-31 lab6-3.o
hxxdrich17@fedora:~/work/arch-pc/lab06$ ./lab6-31
Результат: 5
Остаток от деления: 1
hxxdrich17@fedora:~/work/arch-pc/lab06$
```

Рис. 4.13: Запуск измененной третьей программы

Создаю новый файл и помещаю текст из листинга (рис. -fig. 4.14).



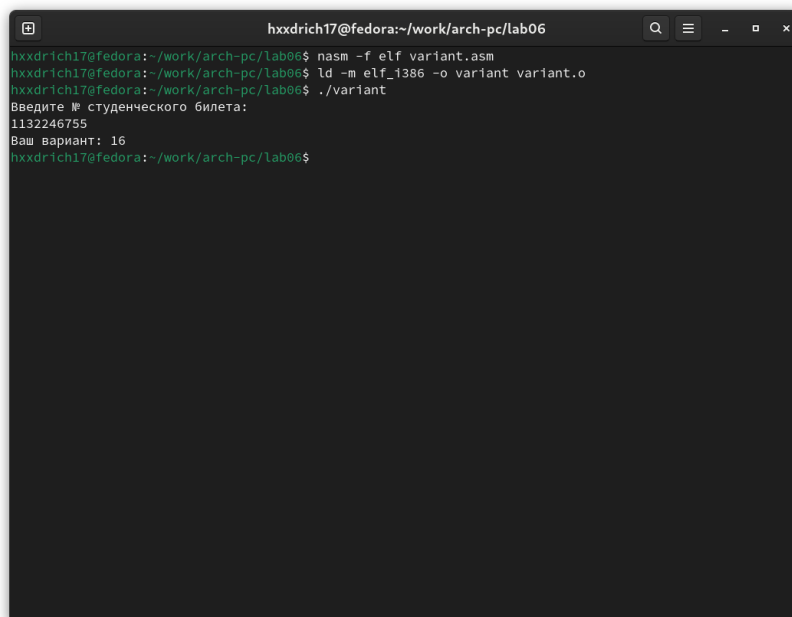
```
variant.asm
~/work/arch-pclab06

port.md | report.md | in_out.asm | report.md | lab6-1.asm | lab6-2.asm | lab6-3.asm | variant.as x

%include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
ren: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, ren
call sprintf
mov eax, edx
call iprintf
call quit
```

Рис. 4.14: Программа для подсчета варианта

Запустив программу и указав свой номер студенческого билета, я получил свой вариант для дальнейшей работы. (рис. -fig. 4.15).



```
hxxdrich17@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
hxxdrich17@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
hxxdrich17@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246755
Ваш вариант: 16
hxxdrich17@fedora:~/work/arch-pc/lab06$
```

Рис. 4.15: Запуск программы для подсчета варианта

4.3 Ответы на контрольные вопросы

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```
mov eax, rem
call sprint
```

2. 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.
3. `call atoi` используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`.
4. 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
```

5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx.
6. Инструкция inc edx увеличивает значение регистра edx на 1.
7. За вывод на экран результатов вычислений отвечают строки:

```
mov eax,edx
call iprintLF
```

4.4 Задание для самостоятельной работы

В соответствии с выбранным вариантом, я реализую программу для подсчета функции $f(x) = (10x-5)^2$, проверка на нескольких переменных показывает корректное выполнение программы (рис. -fig. 4.16).

```
hxxdrich17@fedora:~/work/arch-pc/lab06$ cat lab6-4.asm
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov ebx, 10
mul ebx
sub eax, 5
mul eax
mov edi, eax
mov eax, rem
call sprint
mov eax, edi
call iprintLF
call quit

hxxdrich17@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
hxxdrich17@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
hxxdrich17@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 3
Результат: 625
hxxdrich17@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 25
hxxdrich17@fedora:~/work/arch-pc/lab06$
```

Рис. 4.16: Запуск и проверка программы

Прилагаю код своей программы:

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
```

```
call sread
mov eax, x
call atoi
mov ebx, 10
mul ebx
sub eax, 5
mul eax
mov edi, eax
mov eax, rem
call sprint
mov eax, edi
call iprintLF
call quit
```

5 Выводы

При выполнении данной лабораторной работы я освоил арифметические инструкции языка ассемблера NASM.

6 Список литературы

1. Курс на ТУИС
2. Лабораторная работа №6
3. Программирование на языке ассемблера NASM Столяров А. В.