

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Панина Жанна Валерьевна

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	15
4.2.1	Ответы на вопросы по программе	18
4.3	Выполнение заданий для самостоятельной работы	19
5	Выводы	22
6	Список литературы	23

Список иллюстраций

4.1	Создание файла	9
4.2	Создание копии файла	9
4.3	Редактирование файла	10
4.4	Запуск исполняемого файла	11
4.5	Редактирование файла	11
4.6	Запуск исполняемого файла	12
4.7	Редактирование файла	12
4.8	Запуск исполняемого файла	13
4.9	Редактирование файла	13
4.10	Запуск исполняемого файла	14
4.11	Редактирование файла	14
4.12	Запуск исполняемого файла	15
4.13	Редактирование файла	15
4.14	Запуск исполняемого файла	16
4.15	Изменение программы	16
4.16	Запуск исполняемого файла	17
4.17	Редактирование файла	17
4.18	Запуск исполняемого файла	18
4.19	Написание программы	19
4.20	Запуск исполняемого файла	20
4.21	Запуск исполняемого файла	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.

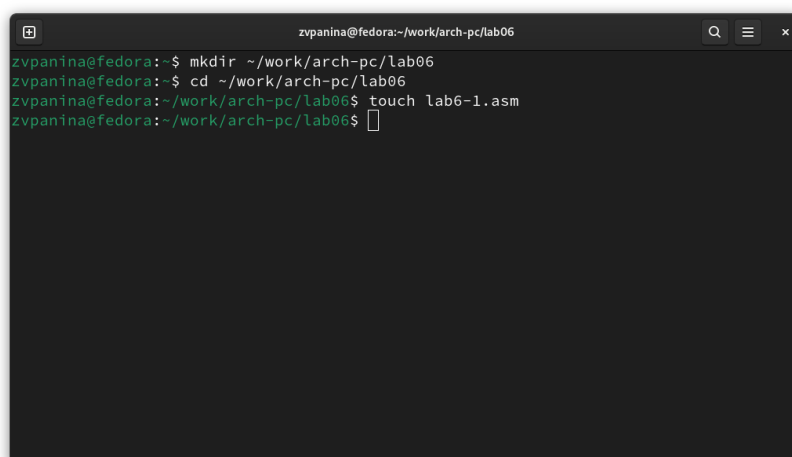
- Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`.
- Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`.
- Адресация памяти – операнд задает адрес в памяти. В команде указывается 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

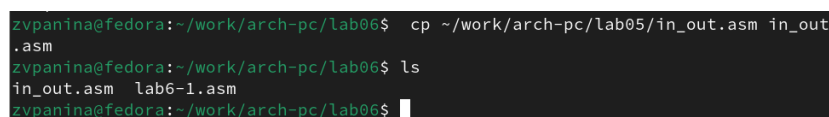
Создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6. Перехожу в созданный каталог с помощью утилиты `cd`. С помощью команды `touch` создаю файл `lab6-1.asm` (рис. 4.1).



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

Рис. 4.1: Создание файла

Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp` (рис. 4.2).

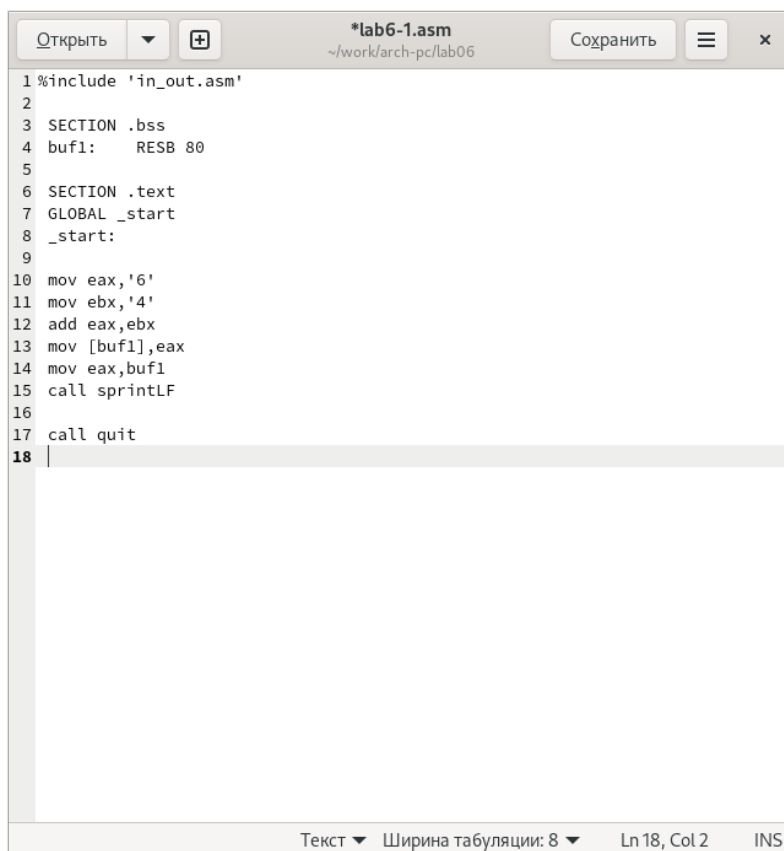


```
zvpanina@fedora:~/work/arch-pc/lab06$ cp ~/work/arch-pc/lab05/in_out.asm in_out
.asm
zvpanina@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm
zvpanina@fedora:~/work/arch-pc/lab06$
```

Рис. 4.2: Создание копии файла

Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода

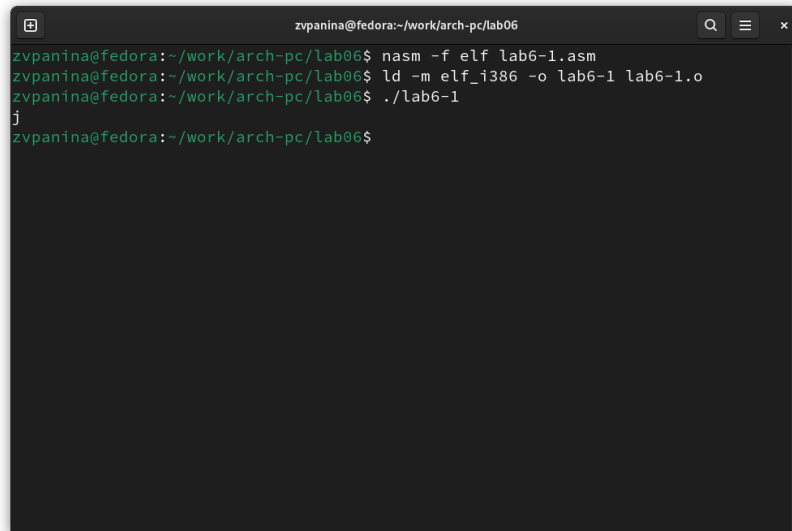
значения регистра еах (рис. 4.3).



```
1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1:  RESB 80
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov eax,'6'
11 mov ebx,'4'
12 add eax,ebx
13 mov [buf1],eax
14 mov eax,buf1
15 call sprintf
16
17 call quit
18 |
```

Рис. 4.3: Редактирование файла

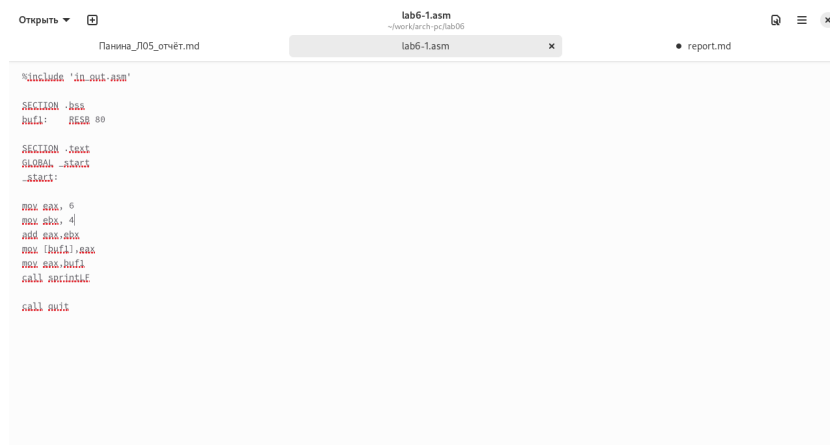
Создаю исполняемый файл программы и запускаю его (рис. 4.4). В данном случае при выводе значения регистра еах мы ожидаем увидеть число 10. Однако результатом будет символ j. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда add еах, ебх запишет в регистр еах сумму кодов – 01101010 (106), что в свою очередь является кодом символа j.



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

Рис. 4.4: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.5).



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

%include 'in-out.asm'

SECTION .bss
buf:    RESB 80

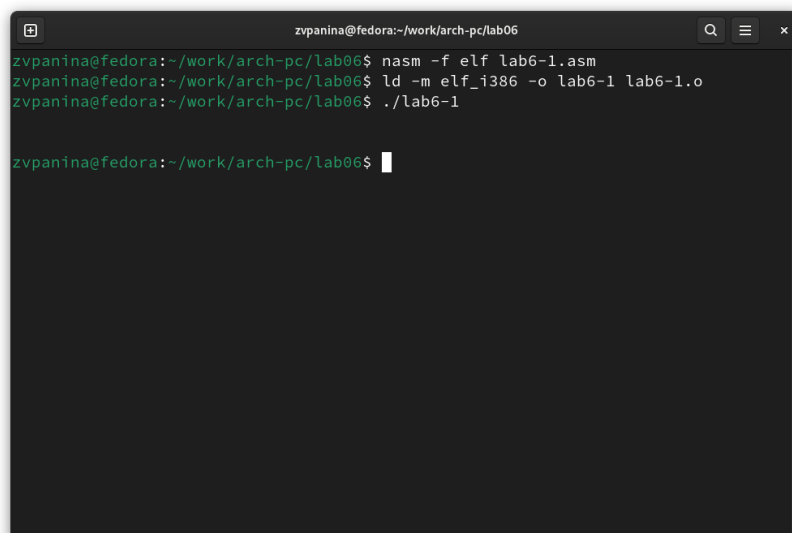
SECTION .text
GLOBAL _start
_start:

    mov eax, 6
    mov ebx, 4
    add eax, ebx
    mov [buf], eax
    mov eax, buf
    call mywrite

    call _start
```

Рис. 4.5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.6). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

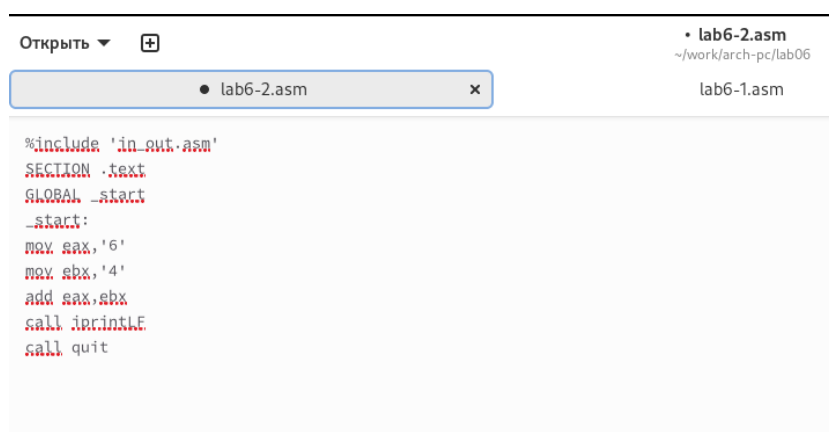


```
zvpanina@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
zvpanina@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
zvpanina@fedora:~/work/arch-pc/lab06$ ./lab6-1

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

Рис. 4.6: Запуск исполняемого файла

Создаю новый файл lab6-2.asm и ввожу в файл текст программы из листинга 6.2 (рис. 4.7).

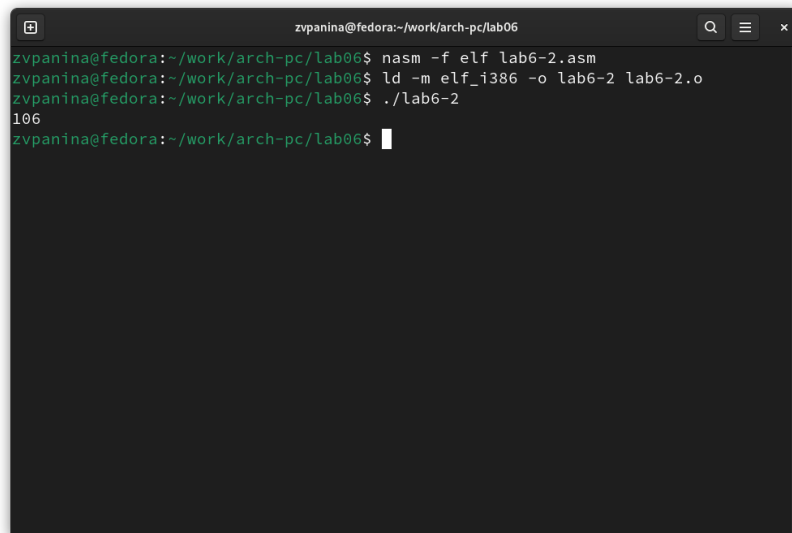


```
Открыть ▾ +
• lab6-2.asm
~ /work/arch-pc/lab06
lab6-1.asm

%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

Рис. 4.7: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 4.8). Теперь выводится число 106, так как функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.



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

Рис. 4.8: Запуск исполняемого файла

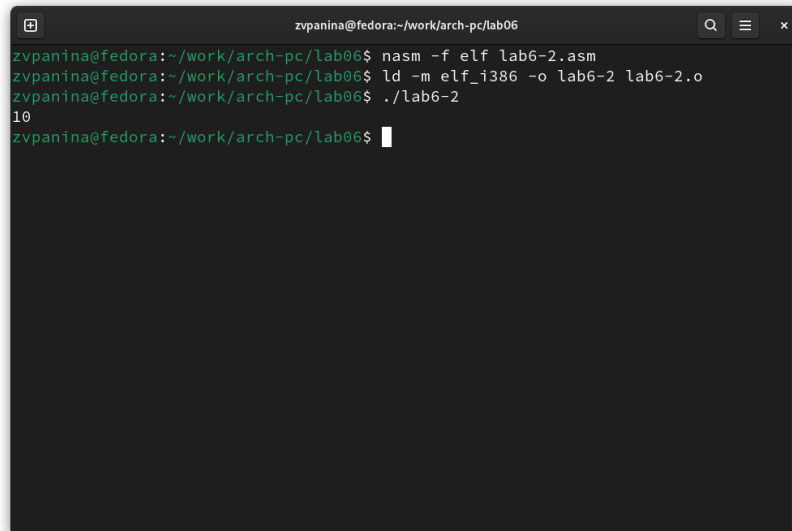
Аналогично предыдущему примеру заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.9).



```
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
```

Рис. 4.9: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.10). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.



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

Рис. 4.10: Запуск исполняемого файла

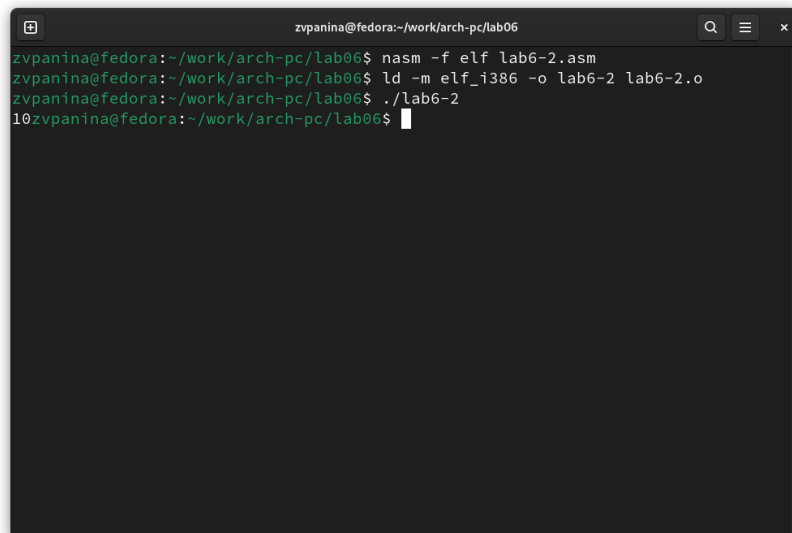
Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. 4.11).



```
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
```

Рис. 4.11: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.12). Вывод теперь осуществляется без переноса строки, т.е. `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF`.

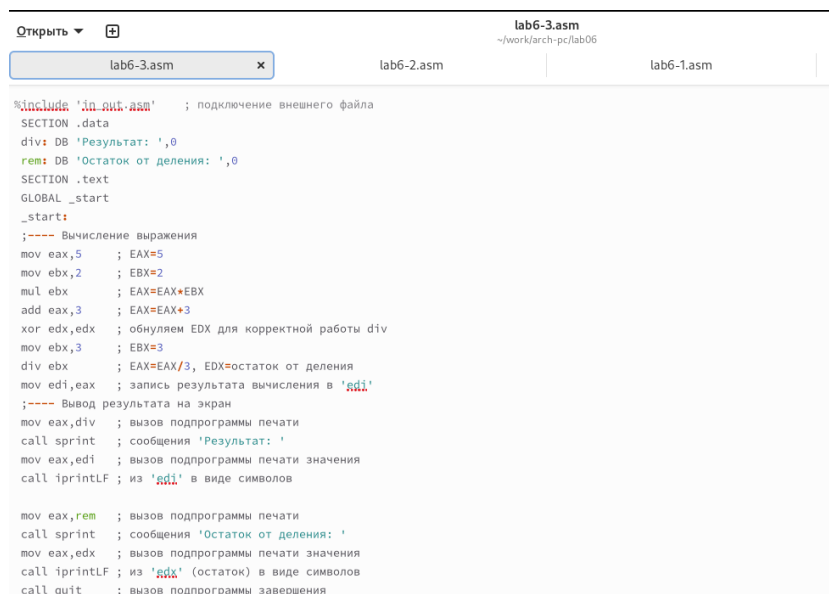


```
zvipanina@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
zvipanina@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
zvipanina@fedora:~/work/arch-pc/lab06$ ./lab6-2
10zvipanina@fedora:~/work/arch-pc/lab06$
```

Рис. 4.12: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm и ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 4.13).



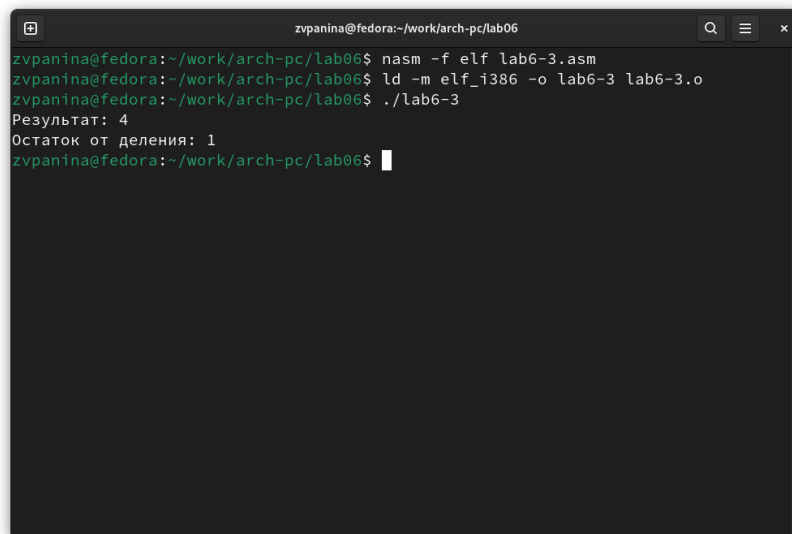
```
Открыть ▾
lab6-3.asm
lab6-2.asm
lab6-1.asm

%include 'lab06.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'
;---- 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
```

Рис. 4.13: Редактирование файла

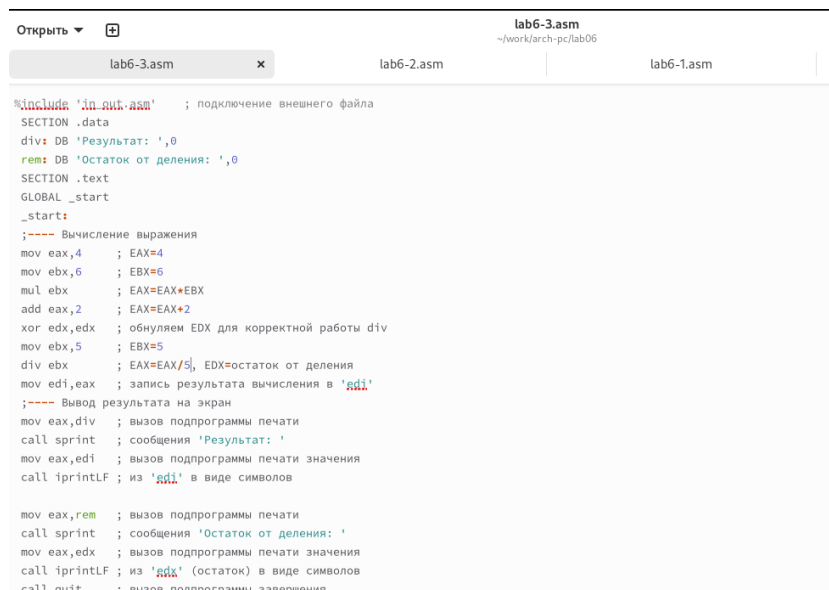
Создаю исполняемый файл и запускаю его (рис. 4.14).



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

Рис. 4.14: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.15).



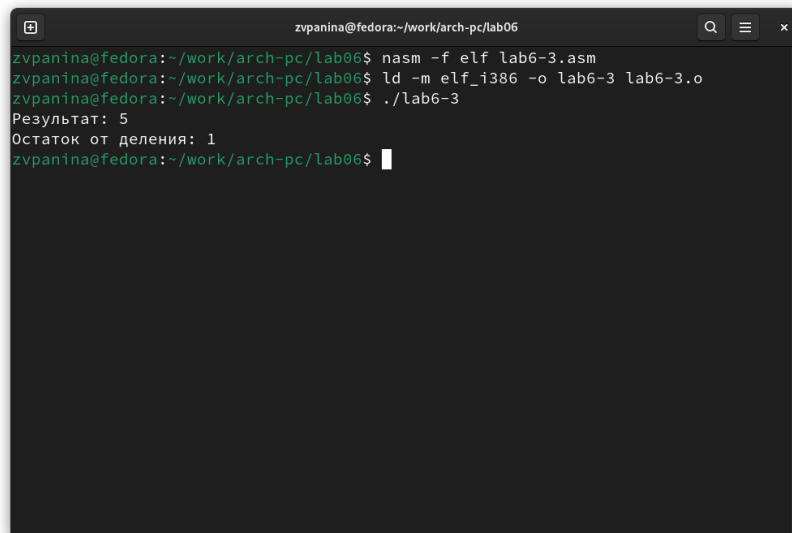
```
lab6-3.asm
~/.work/arch-pc/lab06

%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 'div'
;---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'div' 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 'div' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.15: Изменение программы

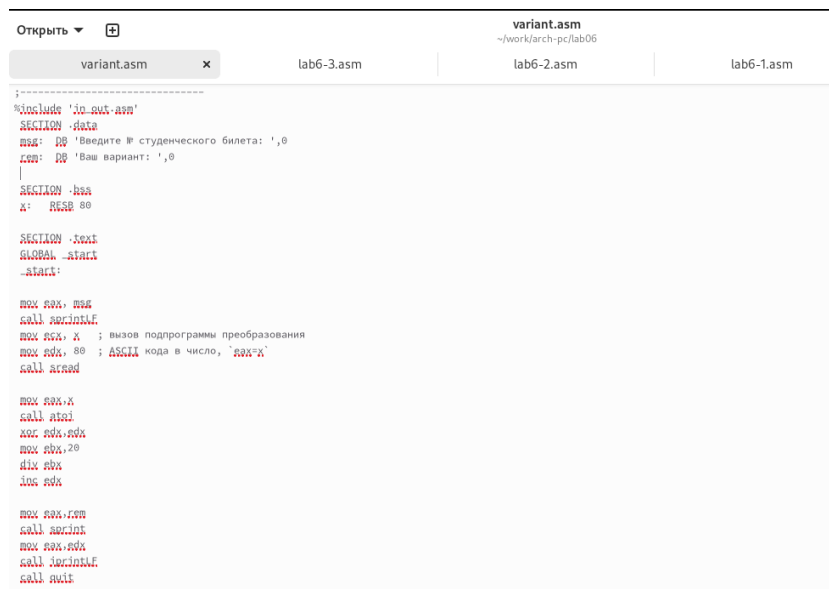
Создаю и запускаю новый исполняемый файл (рис. 4.16). Проверяя значение выражения самостоятельно, понимаю, что программа работает верно.



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

Рис. 4.16: Запуск исполняемого файла

Создаю файл `variant.asm` и ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.17).



```
variant.asm
~/.work/arch-pc/lab06

%include 'io_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
len: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:

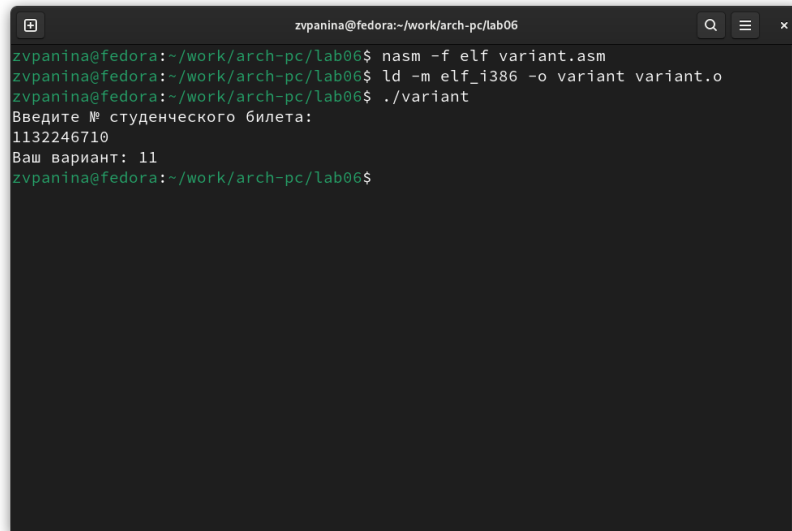
mov eax, msg
call sprintf
mov ecx, x ; вызов подпрограммы преобразования
mov edx, 80 ; ASCII кода в число, 'eax=x'
call atoi

mov eax, x
call atoi
xor edx, edx
mov ebx, 20
div ebx
inc edx

mov eax, ret
call sprintf
mov eax, edx
call sprintf
call write
```

Рис. 4.17: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.18). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант 11.



```
zvipanina@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
zvipanina@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
zvipanina@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246710
Ваш вариант: 11
zvipanina@fedora:~/work/arch-pc/lab06$
```

Рис. 4.18: Запуск исполняемого файла

4.2.1 Ответы на вопросы по программе

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 ; $edx = edx + 1$

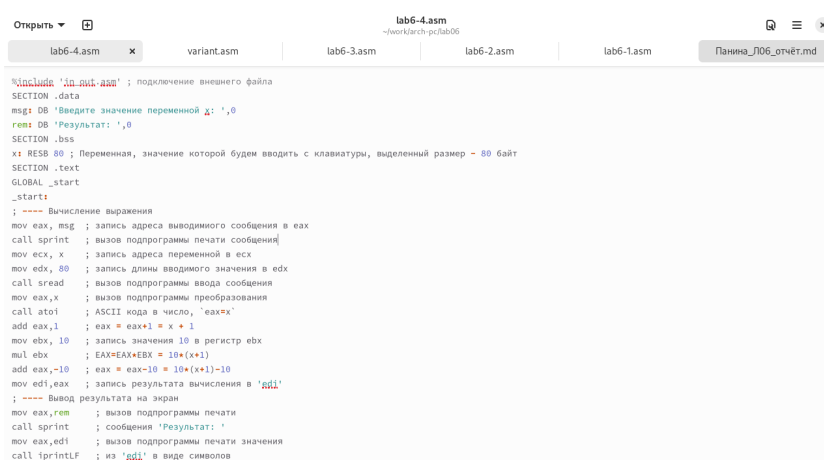
5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1
7. За вывод на экран результатов вычислений отвечают строки:

mov eax,edx

call iprintLF

4.3 Выполнение заданий для самостоятельной работы

Создаю файл `lab6-4.asm` и ввожу в него текст программы для вычисления значения выражения $10 \cdot (x+1) - 10$ (рис. 4.19).



```
lab6-4.asm
~\work\arch\pelab06

lab6-4.asm x variant.asm lab6-3.asm lab6-2.asm lab6-1.asm Панина_Ю06_отчет.md

%include 'in_mml.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, выделенный размер - 80 байт
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в eax
call sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
add eax, 1 ;  $eax = eax + 1$ 
mov ebx, 10 ; запись значения 10 в регистр ebx
mul ebx ;  $EAX=ECX*EBX = 10 \cdot (x+1)$ 
add eax, -10 ;  $eax = eax - 10 = 10 \cdot (x+1) - 10$ 
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
```

Рис. 4.19: Написание программы

Создаю и запускаю исполняемый файл (рис. 4.20). Вычисляю самостоятельно для проверки, результат верный.

```
zvpanina@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
zvpanina@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
zvpanina@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 10
zvpanina@fedora:~/work/arch-pc/lab06$
```

Рис. 4.20: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. 4.21). Программа отработала верно.

```
zvpanina@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 7
Результат: 70
zvpanina@fedora:~/work/arch-pc/lab06$
```

Рис. 4.21: Запуск исполняемого файла

Листинг 4.1. Программа для вычисления значения выражения $10 * (x + 1) - 10$.

```
%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, выделенный размер

SECTION .text

GLOBAL _start

_start:

; ---- Вычисление выражения

mov eax, msg ; запись адреса выводимого сообщения в eax

call sprint ; вызов подпрограммы печати сообщения

mov ecx, x ; запись адреса переменной в ecx

mov edx, 80 ; запись длины вводимого значения в edx

call sread ; вызов подпрограммы ввода сообщения

mov eax, x ; вызов подпрограммы преобразования

call atoi ; ASCII кода в число, 'eax=x'

add eax, 1 ; $eax = eax + 1 = x + 1$

mov ebx, 10 ; запись значения 10 в регистр ebx

mul ebx ; $EAX = EAX * EBX = 10 * (x + 1)$

add eax, -10 ; $eax = eax - 10 = 10 * (x + 1) - 10$

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

1. Лабораторная работа №7
2. Таблица ASCII