

Лабораторная работа №6

Арифметические операции в NASM

Акунаева Антонина Эрдниевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Символьные и численные данные в NASM	7
3.2	Выполнение арифметических операций в NASM	13
4	Описание результатов выполнения заданий для самостоятельной работы	21
5	Выводы	24

Список иллюстраций

3.1	Использование команд mkdir и touch	7
3.2	Midnight Commander. Каталог ~/work/arch-pc/lab06/	8
3.3	МС. Mcedit: lab6-1.asm	9
3.4	Трансляция, компоновка и запуск исполняемого файла lab6-1 . . .	9
3.5	МС. Mcedit: lab6-1 - внесение изменений	10
3.6	Трансляция, компоновка и запуск исполняемого изменённого файла lab6-1	10
3.7	Использование touch	11
3.8	МС. Mcedit: файл lab6-2.asm	11
3.9	Трансляция, компоновка и запуск исполняемого файла lab6-2 . . .	11
3.10	МС. Mcedit: файл lab6-2.asm без кавычек	12
3.11	Трансляция, компоновка и запуск исполняемого файла lab6-2: без кавычек	12
3.12	МС. Mcedit: файл lab6-2.asm с iprint	13
3.13	Трансляция, компоновка и запуск исполняемого файла lab6-2: с iprint	13
3.14	Создание lab6-3.asm при помощи touch	14
3.15	МС. Mcedit: файл lab6-3.asm	15
3.16	Трансляция, компоновка и запуск исполняемого файла lab6-3 . . .	15
3.17	МС. Mcedit: изменённый файл lab6-3.asm	16
3.18	Трансляция, компоновка и запуск изменённого исполняемого файла lab6-3	17
3.19	Создание variant.asm при помощи touch	17
3.20	МС. Mcedit: файл variant.asm	18
3.21	Трансляция, компоновка и запуск исполняемого файла variant . .	19
4.1	Создание var13.asm при помощи touch	21
4.2	МС. Mcedit: файл var13.asm	22
4.3	Трансляция, компоновка и запуск исполняемого файла var13 . . .	23

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

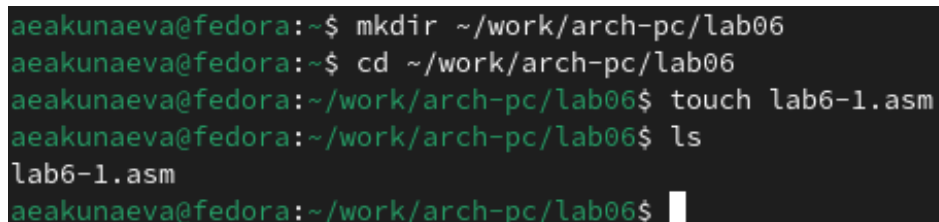
2 Задание

Научиться работать с численными и символьными данными в NASM.
Освоить команды для операций над числами в NASM.

3 Выполнение лабораторной работы

3.1 Символьные и численные данные в NASM

3.1.1. Создайте каталог для программ лабораторной работы №6, перейдите в него и создайте файл lab6-1.asm.



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

Рис. 3.1: Использование команд mkdir и touch

Создадим каталог lab06 в рабочем каталоге при помощи mkdir, перейдём в него с cd. В новом каталоге создадим NASM-файл lab6-1.asm при помощи touch.

3.1.2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax.

Введите в файл lab6-1.asm текст программы из листинга 6.1. Создайте исполняемый файл и запустите его.

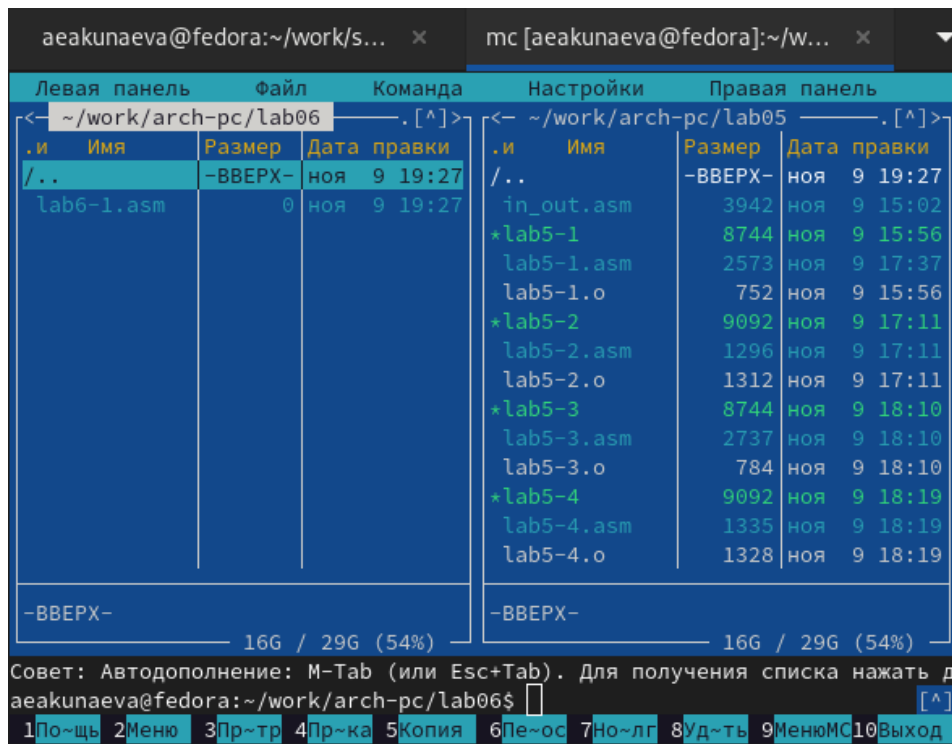
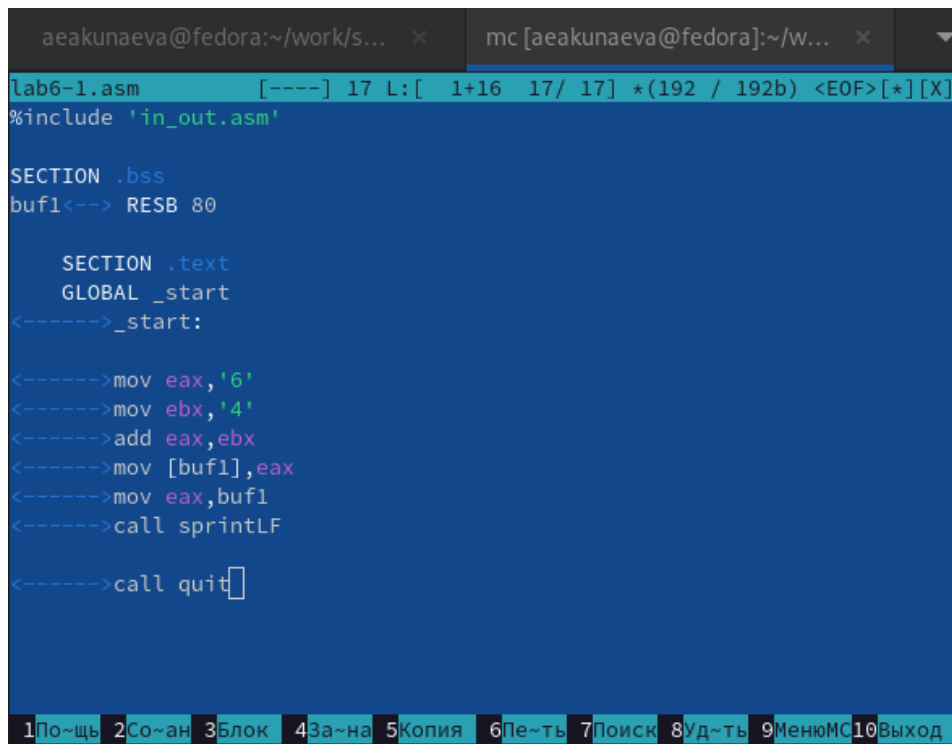


Рис. 3.2: Midnight Commander. Каталог ~/work/arch-pc/lab06/

Откроем Midnight Commander (mc) в текущей директории и функциональной клавишей F4 откроем файл lab6-1.asm в текстовом редакторе mcedit.



```
aeakunaeva@fedora:~/work/s... x mc [aeakunaeva@fedora]:~/w... x
lab6-1.asm [----] 17 L:[ 1+16 17/ 17] *(192 / 192b) <EOF>[*][X]
#include 'in_out.asm'

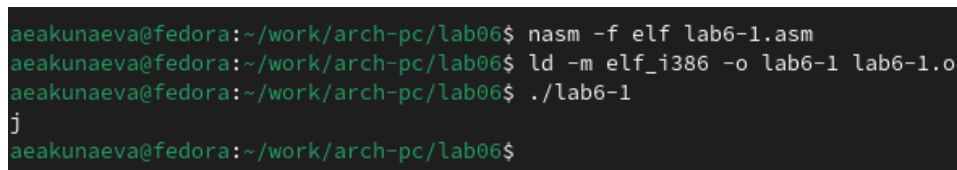
SECTION .bss
buf1<--> RESB 80

SECTION .text
GLOBAL _start
<----->_start:

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

<----->call quit
```

Рис. 3.3: MC. Mcedit: lab6-1.asm



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

Рис. 3.4: Трансляция, компоновка и запуск исполняемого файла lab6-1

Скопируем текст листинга 6.1 в файл и сохраним. Затем оттранслируем, скомпонуем и запустим исполняемый файл lab6-1. Результатом будет один символ 'j', т.к. мы получили его номер в ASCII (всё с предварительным копированием файла in_out.asm в каталог с исполняемым файлом, т.к. в файле обращаемся к нему).

3.1.3. Исправьте текст программы (Листинг 6.1) следующим образом:

замените строки

```
mov eax,'6'
```

```
mov ebx,'4'
```

на строки

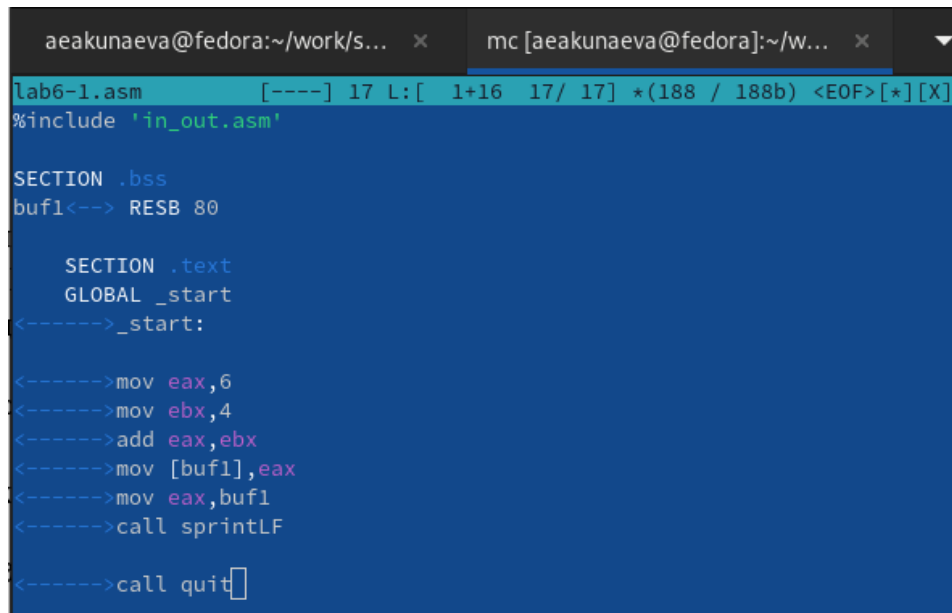
```
mov eax,6
```

```
mov ebx,4
```

Создайте исполняемый файл и запустите его.

Пользуясь таблицей ASCII определите какому символу соответствует код 10.

Отображается ли этот символ при выводе на экран?



```
lab6-1.asm [----] 17 L: [ 1+16 17/ 17] *(188 / 188b) <EOF>[*][X]
#include 'in_out.asm'

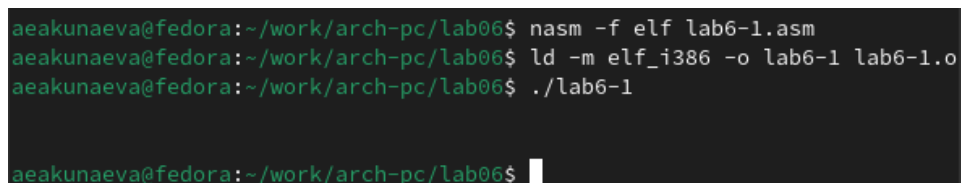
SECTION .bss
buf1<--> RESB 80

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. 3.5: MC. Mcedit: lab6-1 - внесение изменений

Внесём необходимые изменения, убрав кавычки у чисел.



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

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

Рис. 3.6: Трансляция, компоновка и запуск исполняемого изменённого файла lab6-1

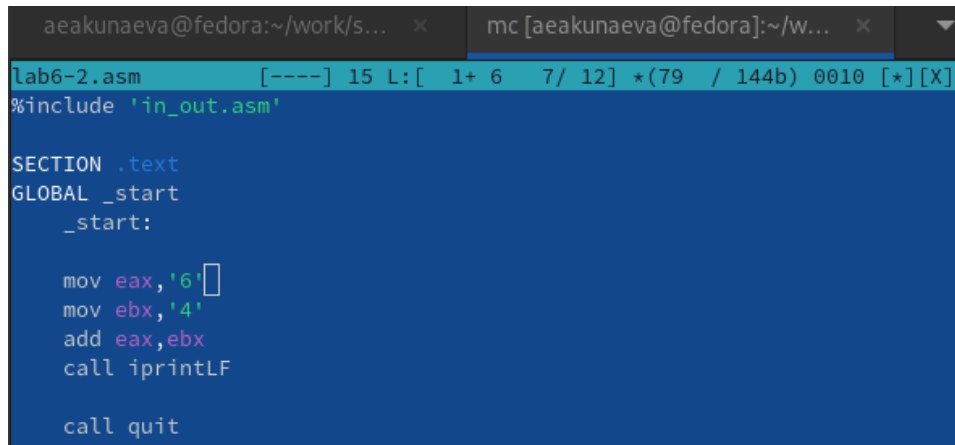
Создадим и запустим новый исполняемый файл lab6-1. Выводится символом с кодом 10 - перевод строки, что подтверждается в таблице ASCII.

3.1.4. Создайте файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введите в него текст программы из листинга 6.2.

```
aeakunaeva@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2
.asm
aeakunaeva@fedora:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2.asm
aeakunaeva@fedora:~/work/arch-pc/lab06$
```

Рис. 3.7: Использование touch

При помощи команды touch создадим файл lab6-2.asm.



```
lab6-2.asm [----] 15 L: [ 1+ 6 7/ 12] *(79 / 144b) 0010 [*] [X]
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

    call quit
```

Рис. 3.8: MC. Mcedit: файл lab6-2.asm

Скопируем данные из листинга 6.2 в файл lab6-2.asm, открыв его в mcedit (Midnight Commander).

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

Рис. 3.9: Трансляция, компоновка и запуск исполняемого файла lab6-2

Снова создадим исполняемый файл с предварительной трансляцией и компоновкой, после запустим. Мы получили число 106.

3.1.5. Аналогично предыдущему примеру изменим символы на числа. Замените строки

```
mov eax,'6'
mov ebx,'4'
```

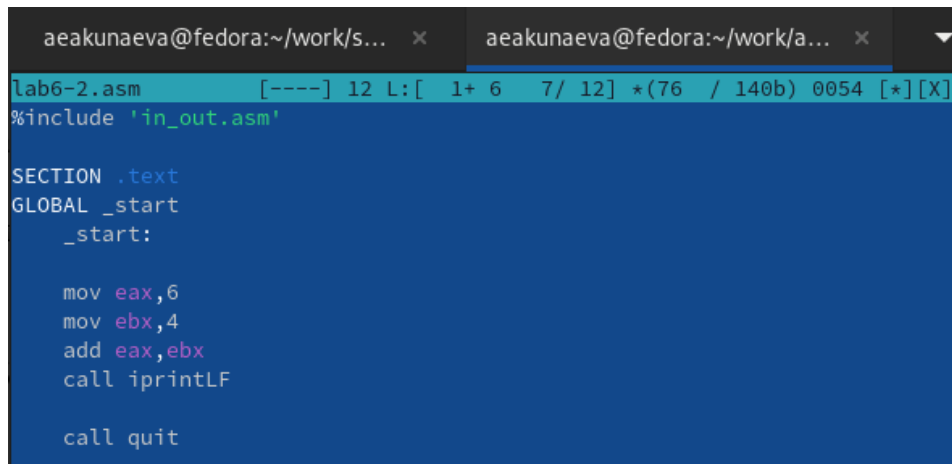
на строки

```
mov eax,6
```

```
mov ebx,4
```

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

Замените функцию `iprintLF` на `iprint`. Создайте исполняемый файл и запустите его. Чем отличается вывод функций `iprintLF` и `iprint`?



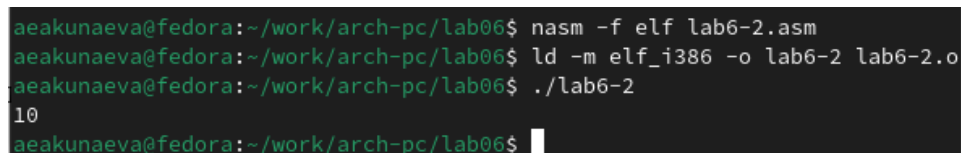
```
aeakunaeva@fedora:~/work/s... x aeakunaeva@fedora:~/work/a... x
lab6-2.asm [----] 12 L:[ 1+ 6 7/ 12] *(76 / 140b) 0054 [*][X]
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

    call quit
```

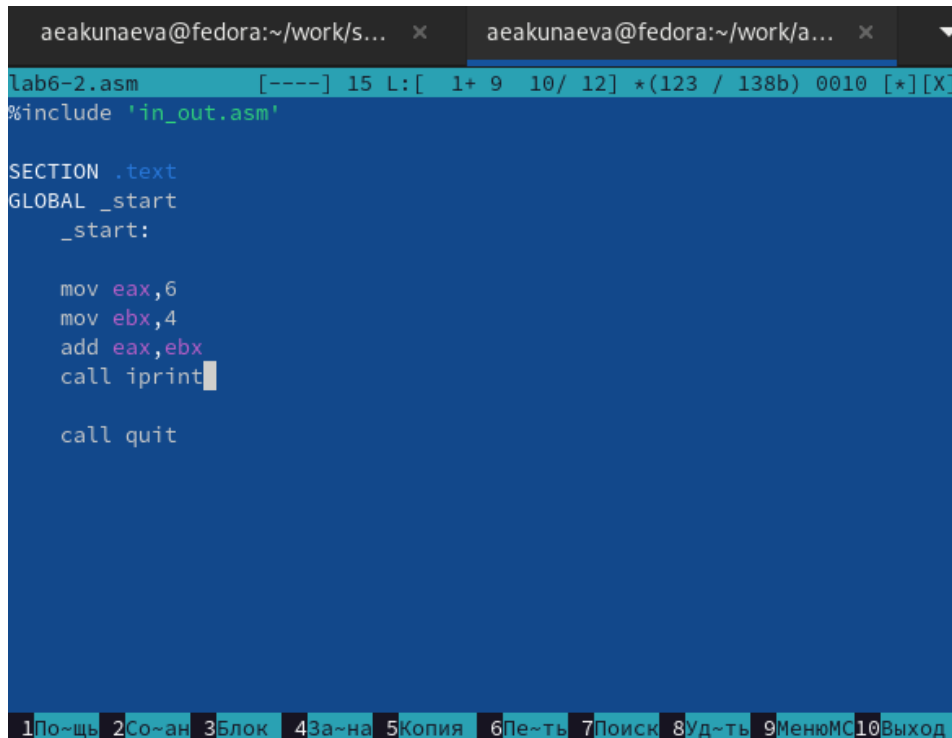
Рис. 3.10: MC. Mcedit: файл lab6-2.asm без кавычек



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

Рис. 3.11: Трансляция, компоновка и запуск исполняемого файла lab6-2: без кавычек

В редакторе проведём необходимые изменения и создадим исполняемый файл, который затем запустим. В результате получаем число 10.



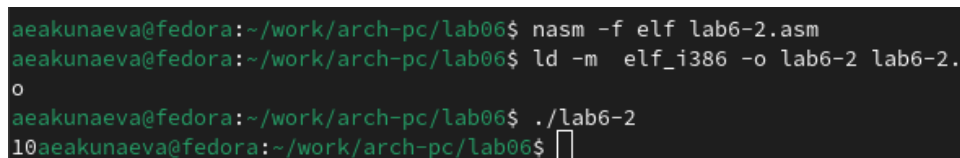
```
aeakunaeva@fedora:~/work/s... x aeakunaeva@fedora:~/work/a... x
lab6-2.asm [----] 15 L: [ 1+ 9 10/ 12] *(123 / 138b) 0010 [*] [X]
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

    call quit
```

Рис. 3.12: MC. Mcedit: файл lab6-2.asm с iprint



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

Рис. 3.13: Трансляция, компоновка и запуск исполняемого файла lab6-2: с iprint

В редакторе проведём необходимые изменения и создадим исполняемый файл, который затем запустим. В результате получаем число 10, но уже без переноса строки в конце, т.к. `iprintLF`, в отличие от `iprint`, сопровождает строку в конце переносом.

3.2 Выполнение арифметических операций в NASM

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

Создайте файл lab6-3.asm в каталоге ~/work/arch-pc/lab06.

Внимательно изучите текст программы из листинга 6.3 и введите в lab6-3.asm.

Создайте исполняемый файл и запустите его. Результат работы программы должен быть следующим:

```
user@dk4n31:~$ ./lab6-3
```

Результат: 4

Остаток от деления: 1

```
user@dk4n31:~$
```

Измените текст программы для вычисления выражения $f(x) = (4 \cdot 6 + 2) / 5$. Создайте исполняемый файл и проверьте его работу.

```
10aeakunaeva@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6
.asm
aeakunaeva@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o
lab6-1      lab6-1.o    lab6-2.asm  lab6-3.asm
aeakunaeva@fedora:~/work/arch-pc/lab06$
```

Рис. 3.14: Создание lab6-3.asm при помощи touch

```
aeakunaeva@fedora:~/work/s... x aeakunaeva@fedora:~/work/a... x
lab6-3.asm [----] 0 L: [ 1+10 11/ 39] *(315 /1469b) 0010 [*] [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,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

1По~щъ 2Со~ан 3Блок 4За~на 5Копия 6Пе~ть 7Поиск 8Уд~ть 9МенюMC10Выход
```

Рис. 3.15: MC. Mcedit: файл lab6-3.asm

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

Рис. 3.16: Трансляция, компоновка и запуск исполняемого файла lab6-3

С помощью команды touch создаём lab6-3.asm. В редакторе введём в него данные из листинга 6.3, предварительно изучив его содержание, и создадим исполняемый файл, который затем запустим. Результат совпадает с описанным в примере.

```

aeakunaeva@fedora:~/work/s... x aeakunaeva@fedora:~/work/a... x
lab6-3.asm [----] 23 L: [ 1+22 23/ 39] *(635 /1469b) 0044 [*] [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/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
1По~щь 2Со~ан 3Блок 4За~на 5Копия 6Пе~ть 7Поиск 8Уд~ть 9МенюMC10Выход

```

Рис. 3.17: MC. Mcedit: изменённый файл lab6-3.asm


```

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

```

Рис. 3.18: Трансляция, компоновка и запуск изменённого исполняемого файла lab6-3

Изменим выражение в редакторе в файле lab6-3.asm соответственно требованиям и запустим заново созданный исполняемый файл. В результате получаем верный для нового выражения ответ.

3.2.2. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

- вывести запрос на введение № студенческого билета
- вычислить номер варианта по формуле: $(S_n \bmod 20) + 1$, где S_n – номер студенческого билета (В данном случае $a \bmod b$ – это остаток от деления a на b).
- вывести на экран номер варианта. В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры.

Создайте файл variant.asm в каталоге ~/work/arch-pc/lab06:

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

Внимательно изучите текст программы из листинга 6.4 и введите в файл variant.asm.

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

```

aeakunaeva@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
aeakunaeva@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm  variant.asm
lab6-1      lab6-1.o   lab6-2.asm  lab6-3    lab6-3.o
aeakunaeva@fedora:~/work/arch-pc/lab06$

```

Рис. 3.19: Создание variant.asm при помощи touch

```
aeakunaeva@fedora:~/work/s... x aeakunaeva@fedora:~/work/a... x
variant.asm [----] 0 L:[ 1+31 32/ 38] *(607 / 673b) 0010 [*][X]
;-----
; Программа вычисления варианта
;-----

%include 'in_out.asm'

SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: 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 ; 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 sprintf
<----->mov eax, edx
<----->call iprintLF

<----->call quit

1По~щъ 2Со~ан 3Блок 4За~на 5Копия 6Пе~ть 7Поиск 8Уд~ть 9МенюМС 10Выход
```

Рис. 3.20: МС. Mcedit: файл variant.asm

```

aeakunaeva@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
aeakunaeva@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
aeakunaeva@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032240492
Ваш вариант: 13
aeakunaeva@fedora:~/work/arch-pc/lab06$

```

Рис. 3.21: Трансляция, компоновка и запуск исполняемого файла variant

С помощью команды touch создаём variant.asm. В редакторе введём в него данные из листинга 6.4, предварительно изучив его содержание, и создадим исполняемый файл, который затем запустим. Результат верен, т.к. остаток от 1032240492 и 20 будет 12, затем $+ 1 = 13$.

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

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

Ответ: за вывод отвечают строки

mov eax,rem - назначение адреса rem по ‘eax’, где rem соответствует строке ‘Ваш вариант:’;

call sprint - вызов подпрограммы вывода в терминал из in_out.asm.

2. Для чего используются следующие инструкции?

mov ecx, x - **Ответ:** назначение адреса x по ‘ecx’, присваивание переменной x введённого с клавиатуры значения (номер студенческого билета);

mov edx, 80 - **Ответ:** указание размера строки ‘edx’ = 80;

call sread - **Ответ:** вызов подпрограммы ввода (считывания) с клавиатуры;

3. Для чего используется инструкция “call atoi”?

Ответ: инструкция “call atoi” используется для вызова подпрограммы преобразования ASCII кода (строки) в целое число, чтобы не вызывать символы по номерам из таблицы ASCII.

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

Ответ: за вычисление варианта отвечают строки

`xor edx,edx` - команда исключающего ИЛИ, обнуляет `edx` (размер строки) перед выполнением деления;

`mov ebx,20` - назначение регистра `ebx` = 20;

`div ebx` - команда беззнакового деления `eax` на `ebx` с записью частного от деления в `eax` и остатка - в `edx`;

`inc edx` - команда инкремента (прибавления единицы) к `edx`.

5. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”?

Ответ: остаток от деления записывается в регистр `edx`.

6. Для чего используется инструкция “`inc edx`”?

Ответ: инструкция “`inc edx`” позволяет прибавить к значению регистра `edx` единицу (инкремент).

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

Ответ: за вывод результата отвечают строки

`mov eax,edx` - назначение регистра `eax` = `edx`;

`call iprintLF` - вызов подпрограммы вывода с переносом строки.

4 Описание результатов выполнения заданий для самостоятельной работы

4.1. 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. Создайте исполняемый файл и проверьте его работу для значений x_1 и x_2 из 6.3.

При выполнении задания преобразовывать (упрощать) выражения для $f(x)$ нельзя. При выполнении деления в качестве результата можно использовать только целую часть от деления и не учитывать остаток (т.е. $5 / 2 = 2$).

```
aeakunaeva@fedora:~/work/arch-pc/lab06$ touch var13.asm
aeakunaeva@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.o    lab6-2.o    lab6-3.o    variant.asm
lab6-1      lab6-2      lab6-3      var13.asm   variant.o
lab6-1.asm  lab6-2.asm  lab6-3.asm  variant
aeakunaeva@fedora:~/work/arch-pc/lab06$
```

Рис. 4.1: Создание var13.asm при помощи touch

С помощью команды touch создаём var13.asm. Из таблицы 6.3 выберем 13-ый вариант: $y(x) = (8 * x + 6) * 10$ при $x_1 = 1$, $x_2 = 4$.

```

aeakunaeva@fedora:~/work/s... x aeakunaeva@fedora:~/work
var13.asm [----] 45 L:[ 1+43 44/ 44] *(1877/1877b
;-----
; Программа вычисления функции. Вариант 13
;-----
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data
    fnc: DB 'y(x) = (8x + 6) * 10 ',0
    msg: DB 'Введите x: ',0
    res: DB 'Ответ: y равен ',0

SECTION .bss
    x: RESB 80

SECTION .text
GLOBAL _start
_start:
; ----- Вывод функции на экран -----
    mov eax, fnc ; вызов подпрограммы печати
    call sprintf ; сообщения с функцией
; ----- Ввод X (с клавиатуры) -----
    mov eax, msg ; вызов подпрограммы печати
    call sprintf ; сообщения 'Введите x: '

    mov ecx, x ; вызов подпрограммы считывания
    mov edx, 80
    call sread ; с клавиатуры значения x
    mov eax, x ; вызов подпрограммы преобразования
    call atoi ; ASCII кода в число, `eax = x`

; ----- Вычисление выражения под функцией -----
    mov ebx, 8 ; EBX = 8
    mul ebx ; EAX = EAX * EBX
    add eax, 6 ; EAX = EAX + 6
    mov ebx, 10 ; EBX = 10
    mul ebx ; EAX = EAX * 10
    mov edi, eax ; запись результата вычисления в 'edi'

; ----- Вывод результата на экран -----
    mov eax, res ; вызов подпрограммы печати
    call sprintf ; сообщения 'Ответ: y равен '
    mov eax, edi ; вызов подпрограммы печати значения
    call iprintLF ; из 'edi' в виде символов


    call quit ; вызов подпрограммы завершения

```

Рис. 4.2: MC. Mcedit: файл var13.asm

В текстовом редакторе (открываем при помощи `mcedit var13.asm`) `mcedit` откроем `var13.asm` и запишем команды согласно заданию.

Программа будет выводить функцию, а также запрашивать `x` с клавиатуры. Затем будут проводиться вычисления (умножение `mul`, сумма `add`). В конце будет выводиться получившееся при введённом `x` значение функции `y`.



```
aeakunaeva@fedora:~/work/arch-pc/lab06$ nasm -f elf var13.asm
aeakunaeva@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o var13 var13.o
aeakunaeva@fedora:~/work/arch-pc/lab06$ ./var13
y(x) = (8x + 6) * 10
Введите x: 1
Ответ: y равен 140
aeakunaeva@fedora:~/work/arch-pc/lab06$ ./var13
y(x) = (8x + 6) * 10
Введите x: 4
Ответ: y равен 380
aeakunaeva@fedora:~/work/arch-pc/lab06$
```

Рис. 4.3: Трансляция, компоновка и запуск исполняемого файла `var13`

Создадим исполняемый файл, предварительно оттранслировав объектный файл и скомпоновав, затем запустим. Введём вместо `x` значения `x1 = 1` и `x2 = 4`. Если проверить аналитически, результат для `x1` и `x2` будет верен.

5 Выводы

Я освоила арифметические инструкции языка ассемблера NASM.