

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Гафоров Нурмухаммад Вомикович

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
4.2.1	Ответы на вопросы по программе	17
4.3	Выполнение заданий для самостоятельной работы	18
5	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	Создание директории	9
4.2	Переходили в каталог	9
4.3	Создание файла	9
4.4	Создание копии файла	10
4.5	Открыли каталог	10
4.6	входили в каталог	11
4.7	Редактирование файла	11
4.8	Открытие файла для просмотра	12
4.9	Запуск исполняемого файла	12
4.10	Редактирование файла	12
4.11	Создание файла	13
4.12	Редактирование файла	13
4.13	Запуск исполняемого файла	13
4.14	Редактирование файла	14
4.15	Запуск исполняемого файла	14
4.16	Редактирование файла	14
4.17	Создание файла	15
4.18	Редактирование файла	15
4.19	Запуск исполняемого файла	15
4.20	Изменение программы	16
4.21	Запуск исполняемого файла	16
4.22	Создание файла	16
4.23	Редактирование файла	17
4.24	Запуск исполняемого файла	17
4.25	Создание файла	18
4.26	Написание программы	19
4.27	Запуск исполняемого файла	19
4.28	Запуск исполняемого файла	19

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

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. [4.1])

```
nvgaforov@dk3n54 ~ $ mkdir ~/work/arch-pc/lab06  
nvgaforov@dk3n54 ~ $
```

Рис. 4.1: Создание директории

Перехожу в созданный каталог с помощью утилиты `cd`. (рис. [4.2])

```
nvgaforov@dk3n54 ~ $ cd ~/work/arch-pc/lab06  
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $
```

Рис. 4.2: Переходили в каталог

С помощью утилиты `touch` создаю файл `lab6-1.asm` (рис. [4.3])

```
nvgaforov@dk3n54 ~ $ cd ~/work/arch-pc/lab06  
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ touch lab6-1.asm  
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $
```

Рис. 4.3: Создание файла

Копирую в текущий каталог файл `in_out.asm`, т.к. он будет использоваться в других программах (рис. [4.4]).

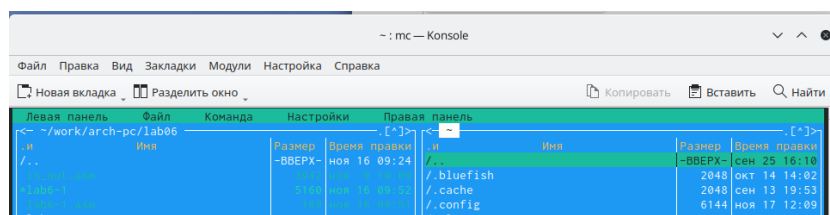


Рис. 4.4: Создание копии файла

с помощью mc откроем созданный файл lab6-1.asm (рис. [4.5]).

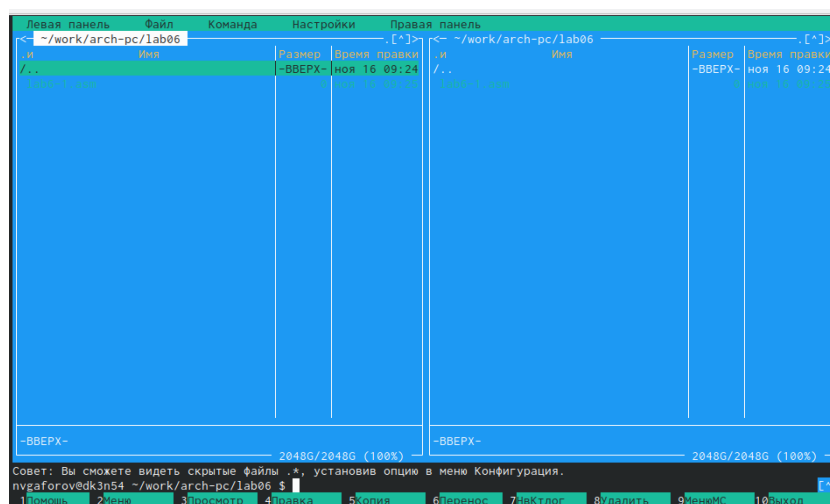


Рис. 4.5: Открыли каталог

Входим в созданный каталог lab6-1.asm (рис. [4.6]).



Рис. 4.6: входили в каталог

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. [4.7]).

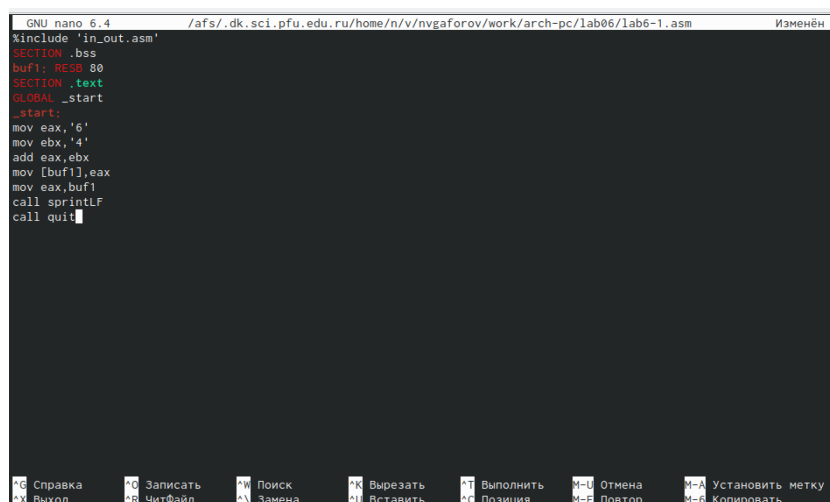


Рис. 4.7: Редактирование файла

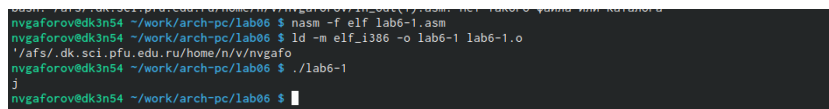
С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы. Открытие файла для просмотра (рис. [4.8]).

A screenshot of a file editor window titled '/afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab06/lab6-1.asm'. The editor has a blue background and a green status bar at the bottom with buttons: 1Помощь, 2Заверн, 3Выход, 4тек, 5Перейти, 6, 7Поиск, 8Исходный, 9Формат, 10Выход. The code in the editor is:

```
%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. 4.8: Открытие файла для просмотра

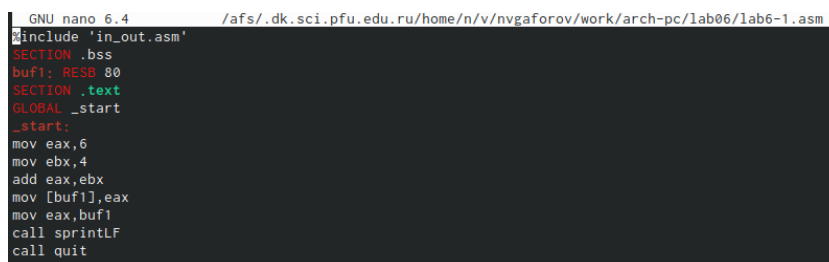
Создаю исполняемый файл программы и запускаю его (рис. [4.9]). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

A screenshot of a terminal window showing the compilation and execution of the assembly program. The commands and output are:

```
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ./lab6-1
j
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $
```

Рис. 4.9: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. [4.10]).

A screenshot of a file editor window titled '/afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab06/lab6-1.asm'. The editor has a dark background and a green status bar at the bottom. The code in the editor is:

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab06/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
```

Рис. 4.10: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. [??]). Теперь

вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

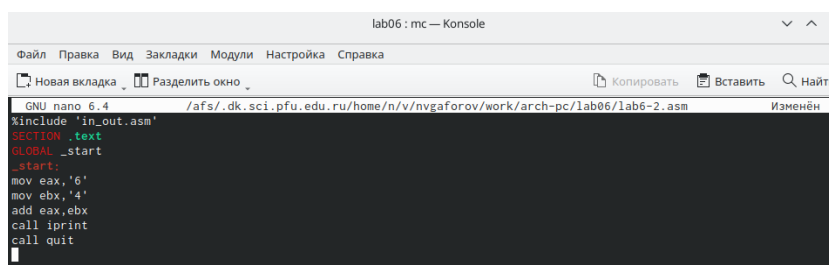
```
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ./lab6-1
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $
```

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. [4.11]).

```
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $
```

Рис. 4.11: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. [4.12]).



```
lab06: mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
[+] Новая вкладка  [ ] Разделить окно  [ ] Копировать  [ ] Вставить  [ ] Найти
/afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab06/lab6-2.asm  Изменён
GNU nano 6.4
#include "in_out.asm"
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprint
call quit
```

Рис. 4.12: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. [4.13]). 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”.

```
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ./lab6-2
106
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $
```

Рис. 4.13: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. [4.14]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab06/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. 4.14: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. [4.15]). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ./lab6-2
10
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $
```

Рис. 4.15: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. [4.16]).

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

Рис. 4.16: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. [??]). 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.

Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. [4.17]).

```
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $
```

Рис. 4.17: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. [4.18]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab06/lab6-3.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,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 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

^G Справка      ^O Записать     ^W Поиск        ^K Вырезать     ^T Выполнить    M-U Отмена      M-A Установить
^X Выход        ^R ЧитФайл     ^N Замена      ^U Вставить     ^C Позиция      M-E Повтор      M-B Копировать
```

Рис. 4.18: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. [4.19]).

```
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $
```

Рис. 4.19: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. [4.20]).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab06/lab6-3.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, EDI=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 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. 4.20: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. [4.21]). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```

nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $

```

Рис. 4.21: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. [4.22]).

```

nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $

```

Рис. 4.22: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. [4.23]).

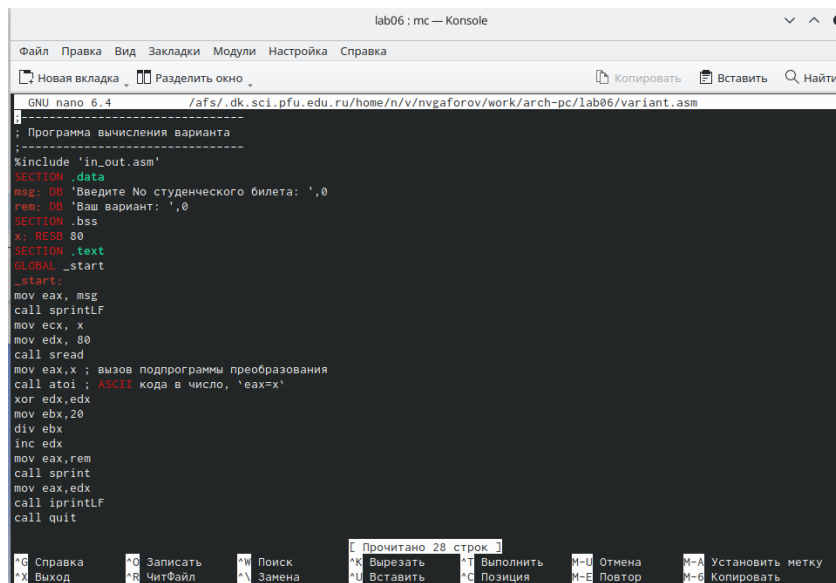


Рис. 4.23: Редактирование файла

Создаю и запускаю исполняемый файл (рис. [4.24]). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 5

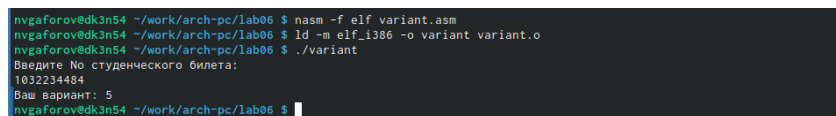


Рис. 4.24: Запуск исполняемого файла

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`

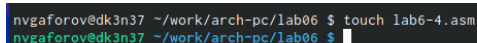
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1

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` с помощью утилиты `touch` (рис. [4.25]).



```
nvgaforov@dk3n37 ~/work/arch-pc/lab06 $ touch lab6-4.asm
nvgaforov@dk3n37 ~/work/arch-pc/lab06 $
```

Рис. 4.25: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(9x-8)/8$ (рис. [4.26]). Это выражение было под вариантом 5.

```
GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab06/lab6-4.asm
#include "in_out.asm" ; подключение внешнего файла
SECTION .data ; секция инициализированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss ; секция не инициализированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; ---- 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 выводимого значения в edx
call spread ; вызов подпрограммы ввода сообщения
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
add eax,8x; eax = 9x-8 = 9x-8
mov ebx,8 ; запись значения 8 в регистр ebx
mul ebx; EAX=EAX/EBX = (9x-8)/8
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 iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.26: Написание программы

Создаю и запускаю исполняемый файл (рис. [4.27]). При вводе значения 8, вывод - 136.

```
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 8
Результат: 136nvgaforov@dk3n54 ~/work/arch-pc/lab06 $
```

Рис. 4.27: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. [4.28]). Программа отработала верно.

```
Введите значение переменной x: 64
Результат: 136nvgaforov@dk3n54 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 64
Результат: 584nvgaforov@dk3n54 ~/work/arch-pc/lab06 $
```

Рис. 4.28: Запуск исполняемого файла

5 Выводы

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

Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. 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/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.

13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. 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.
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
18. — 1120 с. — (Классика Computer Science).
19. Таблица ASCII