

Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Калашникова Ольга Сергеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Символьные и численные данные в NASM	6
2.2	Выполнение арифметических операций в NASM	15
3	Задание для самостоятельной работы	24
4	Выводы	27

Список иллюстраций

2.1	Создание папки, её открытие и создание файла	6
2.2	Проверка наличия папки	6
2.3	Проверка наличия файла	7
2.4	Текст программы	8
2.5	Перемещение файла “in_out.asm”	8
2.6	Создание и запуск исполняемого файла	9
2.7	Изменённый текст программы	10
2.8	Создание и запуск исполняемого файла	10
2.9	Символ 10 в системе ASCII	11
2.10	Создание файла	11
2.11	Проверка наличия файла	11
2.12	Текст программы	12
2.13	Создание и запуск исполняемого файла	12
2.14	Изменённый текст программы	13
2.15	Создание и запуск исполняемого файла	13
2.16	Изменённый текст программы	14
2.17	Создание и запуск исполняемого файла	14
2.18	Создание файла	15
2.19	Проверка наличия файла	16
2.20	Текст программы	17
2.21	Создание и запуск исполняемого файла	17
2.22	Изменённый текст программы	18
2.23	Создание и запуск исполняемого файла	18
2.24	Создание файла	19
2.25	Проверка наличия файла	20
2.26	Текст программы	21
2.27	Создание и запуск исполняемого файла	21
3.1	Текст программы	24
3.2	Создание и запуск исполняемого файла	25

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

2 Выполнение лабораторной работы

2.1 Символьные и численные данные в NASM

Создаём каталог для программ лабораторной работы № 6 (при помощи команды `mkdir ~/work/arch-pc/lab06`), переходим в него (при помощи команды `cd ~/work/arch-pc/lab06`) и создаём файл `lab6-1.asm` (при помощи команды `touch lab6-1.asm`)(рис. 2.1),(рис. 2.2),(рис. 2.2)

```
oskashnikova@dk6n62:~$ mkdir ~/work/arch-pc/lab06
oskashnikova@dk6n62:~$ cd ~/work/arch-pc/lab06
oskashnikova@dk6n62:~/work/arch-pc/lab06$ touch lab6-1.asm
oskashnikova@dk6n62:~/work/arch-pc/lab06$
```

Рис. 2.1: Создание папки, её открытие и создание файла

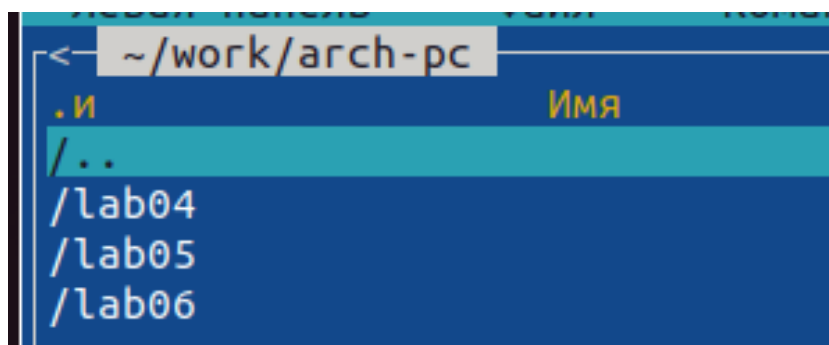


Рис. 2.2: Проверка наличия папки

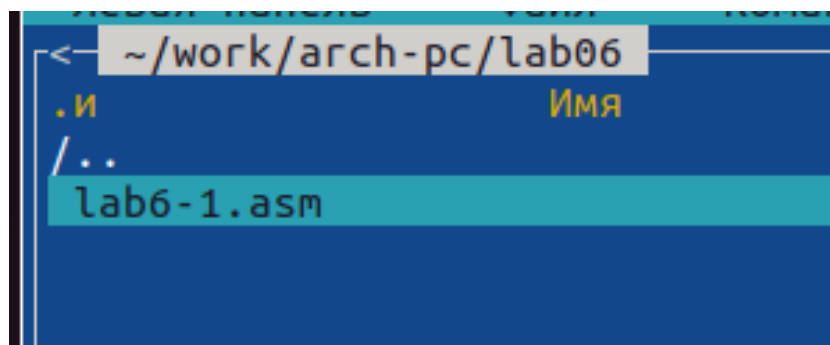


Рис. 2.3: Проверка наличия файла

Введём в файл `lab6-1.asm` текст программы из листинга 6.1., для корректной работы нужно переместить файл `in_out.asm` в тот же каталог, где лежит и файл с текстом программы.(рис. 2.4),(рис. 2.5)

```

/home/oskalashnikova/w
#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. 2.4: Текст программы

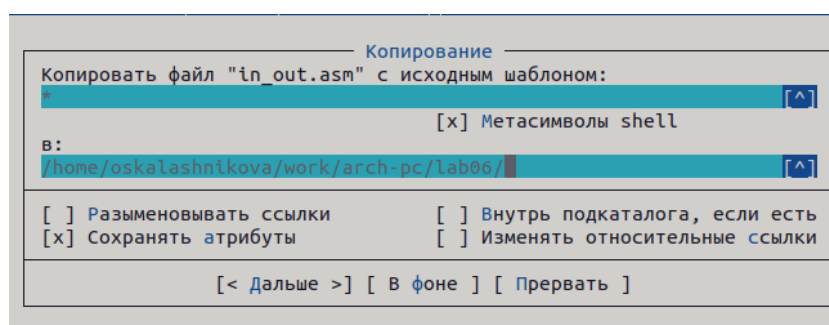


Рис. 2.5: Перемещение файла "in_out.asm"

Создаем исполняемый файл и запускаем его (компиляция: `nasm -f elf lab6-1.asm`, `ld -m elf_i386 -o lab6-1 lab6-1.o`, запуск: `./lab6-1`) (рис. 2.6)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ ./lab6-1
j
oskalashnikova@dk6n62:~/work/arch-pc/lab06$
```

Рис. 2.6: Создание и запуск исполняемого файла

Программа выводит символ `j`, потому что этот символ соответствует по системе ASCII сумме двоичных кодов символов 4 и 6 ($52+54=106$).

Меняем текст программы (вместо символов, записываем в регистры числа) (рис. 2.7)

```

/home/oskashnikova/w
#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. 2.7: Изменённый текст программы

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab6-1-2.asm`, `ld -m elf_i386 -o lab6-1-2 lab6-1-2.o`, запуск: `./lab6-1-2`) (рис. 2.8)

```

oskashnikova@dk6n62:~/work/arch-pc/lab06$ nasm -f elf lab6-1-2.asm
oskashnikova@dk6n62:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1-2 lab6-1-2.o
oskashnikova@dk6n62:~/work/arch-pc/lab06$ ./lab6-1-2

oskashnikova@dk6n62:~/work/arch-pc/lab06$ █

```

Рис. 2.8: Создание и запуск исполняемого файла

Программа выполняет перевод строки, потому что символ перевода строки соответствует по системе ASCII сумме двоичных кодов чисел 4 и 6 ($4+6=10$). (рис. 2.9)

10	12	0x0A	1010	LF, \n
----	----	------	------	--------

Рис. 2.9: Символ 10 в системе ASCII

Создаём файл “lab6-2.asm” в каталоге ~/work/arch-pc/lab06 (touch ~/work/arch-pc/lab06/lab6-2.asm) (рис. 2.10), (рис. 2.10)

```
oskashnikova@dk6n62:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
oskashnikova@dk6n62:~/work/arch-pc/lab06$
```

Рис. 2.10: Создание файла

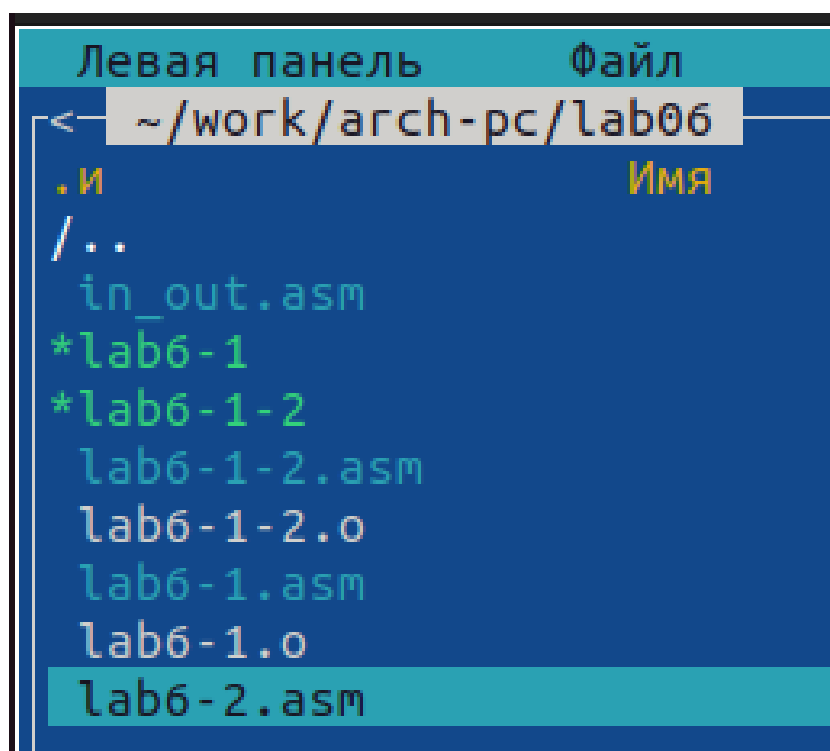


Рис. 2.11: Проверка наличия файла

Вводим в файл текст другой программы для вывода значения регистра еах, которая использует `iprintLF` (рис. 2.12)

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

Рис. 2.12: Текст программы

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab6-2.asm`, `ld -m elf_i386 -o lab6-2 lab6-2.o`, запуск: `./lab6-2`) (рис. 2.13)

```
oskashnikova@dk6n62:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
oskashnikova@dk6n62:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
oskashnikova@dk6n62:~/work/arch-pc/lab06$ ./lab6-2
106
oskashnikova@dk6n62:~/work/arch-pc/lab06$ █
```

Рис. 2.13: Создание и запуск исполняемого файла

Получаем число 106. Команда `add` складывает коды символов '6' и '4' ($54+52=106$). В отличие от программы из листинга 6.1, функция `iprintLF` преобразует код в число.

Заменяем в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 2.14)

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

Рис. 2.14: Изменённый текст программы

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab6-2-2.asm`, `ld -m elf_i386 -o lab6-2-2 lab6-2-2.o`, запуск: `./lab6-2-2`) (рис. 2.15)

```
oskashnikova@dk6n62:~/work/arch-pc/lab06$ nasm -f elf lab6-2-2.asm
oskashnikova@dk6n62:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2-2 lab6-2-2.o
oskashnikova@dk6n62:~/work/arch-pc/lab06$ ./lab6-2-2
10
oskashnikova@dk6n62:~/work/arch-pc/lab06$
```

Рис. 2.15: Создание и запуск исполняемого файла

Получаем число 10. Складываем 6 и 4 ($6+4=10$), с помощью функции `iprintLF`, преобразуем код в число и выводим его на экран.

Заменяем в тексте программы в файле lab6-2-2.asm `iprintLF` на `iprint` (рис. 2.16)

```

/home/oskashnikova/w
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit

```

Рис. 2.16: Изменённый текст программы

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab6-2-3.asm`, `ld -m elf_i386 -o lab6-2-3 lab6-2-3.o`, запуск: `./lab6-2-3`) (рис. 2.17)

```

oskashnikova@dk6n62:~/work/arch-pc/lab06$ nasm -f elf lab6-2-3.asm
oskashnikova@dk6n62:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2-3 lab6-2-3.o
oskashnikova@dk6n62:~/work/arch-pc/lab06$ ./lab6-2-3
10oskashnikova@dk6n62:~/work/arch-pc/lab06$ █

```

Рис. 2.17: Создание и запуск исполняемого файла

`iprint` не выполняет перевод строки, а только выводит число на экран, а `iprintLF` вызывает `iprint` и добавляет символ перевода строки

2.2 Выполнение арифметических операций в NASM

Создаём файл “lab6-3.asm” в каталоге ~/work/arch-pc/lab06 (touch ~/work/arch-pc/lab06/lab6-3.asm) (рис. 2.18), (рис. 2.19)

```
oskashnikova@dk6n62:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm  
oskashnikova@dk6n62:~/work/arch-pc/lab06$
```

Рис. 2.18: Создание файла

```
< ~/work/arch-pc/lab06
.И      Имя
/..
  in_out.asm
*lab6-1
*lab6-1-2
  lab6-1-2.asm
  lab6-1-2.o
  lab6-1.asm
  lab6-1.o
*lab6-2
*lab6-2-2
  lab6-2-2.asm
  lab6-2-2.o
*lab6-2-3
  lab6-2-3.asm
  lab6-2-3.o
  lab6-2.asm
  lab6-2.o
lab6-3.asm
```

Рис. 2.19: Проверка наличия файла

Вводим в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 2.20)


```

/home/oskalashnikova/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 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. 2.20: Текст программы

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab6-3.asm`, `ld -m elf_i386 -o lab6-3 lab6-3.o`, запуск: `./lab6-3`) (рис. 2.21)

```

oskalashnikova@dk6n62:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ █

```

Рис. 2.21: Создание и запуск исполняемого файла

Изменяем программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2) / 5$ (меняются только цифры) (рис. 2.22)

```

/home/oskalashnikova/work/arch-pc/lab06/lab6-3-2.asm [-M--
; -----
; 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=5
mov ebx,6 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; 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. 2.22: Изменённый текст программы

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab6-3-2.asm`, `ld -m elf_i386 -o lab6-3-2 lab6-3-2.o`, запуск: `./lab6-3-2`) (рис. 2.23)

```

oskalashnikova@dk6n62:~/work/arch-pc/lab06$ nasm -f elf lab6-3-2.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3-2 lab6-3-2.o
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ ./lab6-3-2
Результат: 5
Остаток от деления: 1
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ █

```

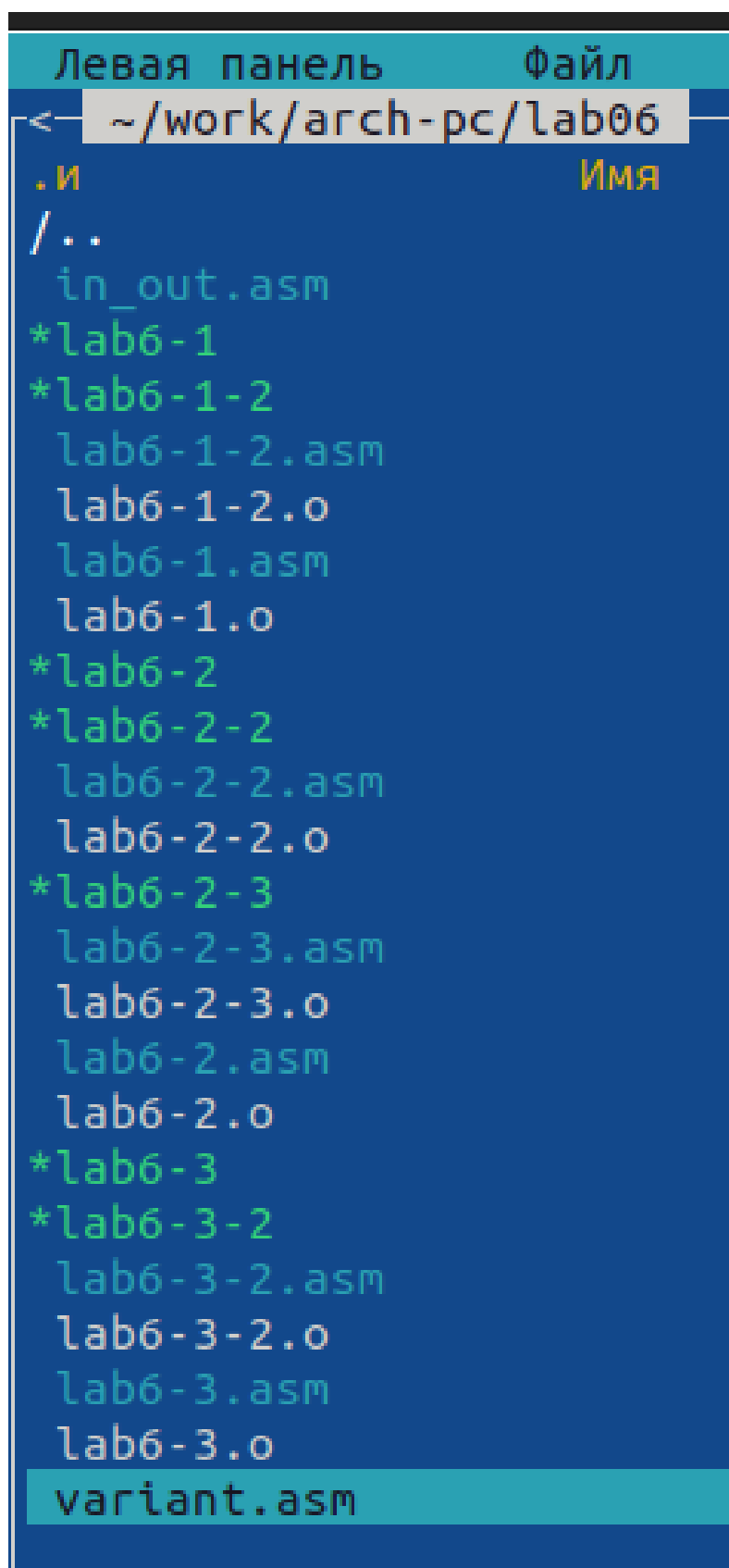
Рис. 2.23: Создание и запуск исполняемого файла

Считаем значения вручную и приходим к выводу, что программа работает верно

Создаём файл “variant.asm” в каталоге `~/work/arch-pc/lab06` (`touch ~/work/arch-pc/lab06/variant.asm`) (рис. 2.24), (рис. 2.25)

```
oskashnikova@dk6n62:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm  
oskashnikova@dk6n62:~/work/arch-pc/lab06$ █
```

Рис. 2.24: Создание файла



The image shows a file manager window with a dark blue background. At the top, there are two tabs: "Левая панель" (Left panel) and "Файл" (File). Below the tabs, the current directory path is displayed as "~/work/arch-pc/lab06". The main area shows a list of files and directories. The first line is ".и" with the word "Имя" (Name) to its right. Below it is a directory entry "/..". Then, there is a list of files and directories, some of which are grouped with an asterisk: "*lab6-1", "*lab6-1-2", "lab6-1-2.asm", "lab6-1-2.o", "lab6-1.asm", "lab6-1.o", "*lab6-2", "*lab6-2-2", "lab6-2-2.asm", "lab6-2-2.o", "*lab6-2-3", "lab6-2-3.asm", "lab6-2-3.o", "lab6-2.asm", "lab6-2.o", "*lab6-3", "*lab6-3-2", "lab6-3-2.asm", "lab6-3-2.o", "lab6-3.asm", "lab6-3.o", and finally "variant.asm" which is highlighted with a light blue background.

```
Левая панель    Файл
< ~/work/arch-pc/lab06
.и              Имя
/..
  in_out.asm
*lab6-1
*lab6-1-2
  lab6-1-2.asm
  lab6-1-2.o
  lab6-1.asm
  lab6-1.o
*lab6-2
*lab6-2-2
  lab6-2-2.asm
  lab6-2-2.o
*lab6-2-3
  lab6-2-3.asm
  lab6-2-3.o
  lab6-2.asm
  lab6-2.o
*lab6-3
*lab6-3-2
  lab6-3-2.asm
  lab6-3-2.o
  lab6-3.asm
  lab6-3.o
variant.asm
```

Рис. 2.25: Проверка наличия файла

Вводим в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 2.26)

```
/home/oskalashnikova/work/arch-pc/lab06/variant.  
;-----  
; 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 iprintf  
call quit
```

Рис. 2.26: Текст программы

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf variant.asm`, `ld -m elf_i386 -o variant variant.o`, запуск: `./variant`) (рис. 2.27)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ nasm -f elf variant.asm  
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o  
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ ./variant  
Введите № студенческого билета:  
1132231846  
Ваш вариант: 7  
oskalashnikova@dk6n62:~/work/arch-pc/lab06$ █
```

Рис. 2.27: Создание и запуск исполняемого файла

При вводе номера своего студенческого я получила 7 вариант

Ответы на вопросы:

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

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

(Так как rem содержит ‘Ваш вариант:’, а call sprint отвечает за вывод на экран сообщения) 2) Для чего используются следующие инструкции?

```
mov ecx, x  
mov edx, 80  
call sread
```

Они используются для того, чтобы ввести значение x с клавиатуры (mov ecx, x помещает адрес x в регистр ecx; mov edx, 80 записывает в регистр edx длину строки; call sread вызывает функцию для ввода сообщения с клавиатуры) 3) Для чего используется инструкция “call atoi”? atoi преобразует ascii-код символа в целое число и записывает результат в регистр eax, call atoi вызывает данную функцию 4) Какие строки листинга 6.4 отвечают за вычисления варианта?

```
xor edx, edx  
mov ebx, 20  
div ebx  
inc edx
```

(xor edx,edx обнуляет edx для корректной работы div; mov ebx,20 записывает 20 в регистр ebx; div ebx производит деление $eax = eax/20$ и записывает остаток в edx; inc edx прибавляет к остатку единицу, чтобы не было 0 варианта) 5) В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”? В регистр edx 6) Для чего используется инструкция “inc edx”? Для увеличения

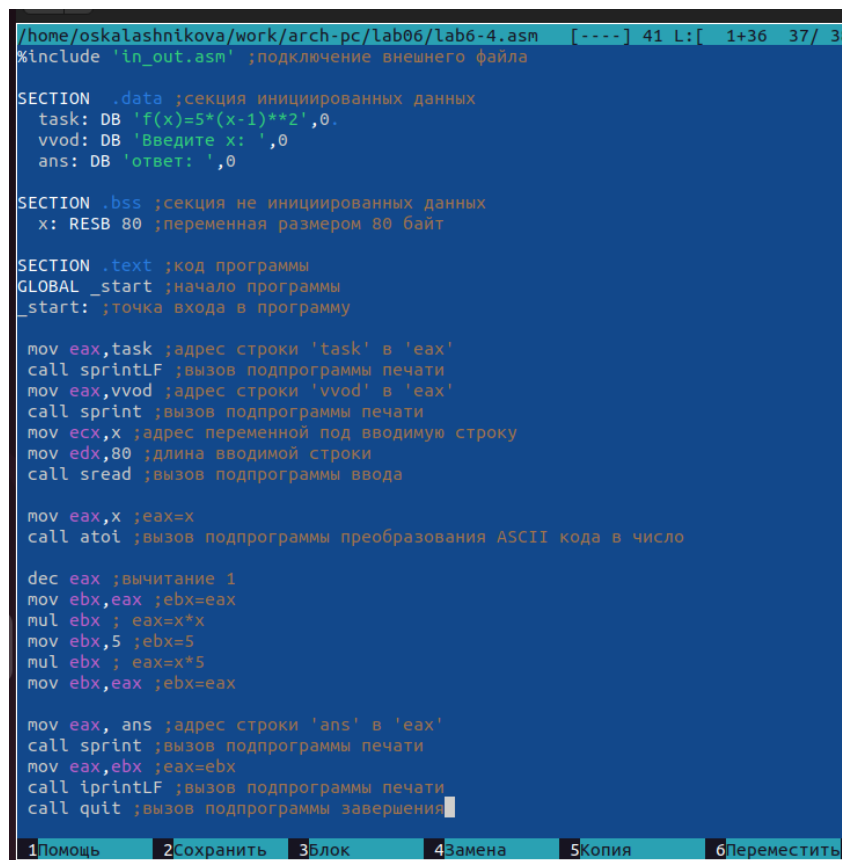
значения `edx` на единицу 7) Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

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

(`mov eax,edx` записывает в `eax` значение вычисленное в `edx`; `call iprintLF` вызывает функцию, которая выводит ответ преобразовав его в число)

3 Задание для самостоятельной работы

Открываем созданный файл lab6-4.asm, вводим в него текст программы для вычисления значения выражения $f(x)=5*(x-1)**2$ (рис. 3.1)



```
/home/oskalashnikova/work/arch-pc/lab06/lab6-4.asm [----] 41 L: [ 1+36 37/ 3
#include 'in_out.asm' ;подключение внешнего файла

SECTION .data ;секция иницированных данных
task: DB 'f(x)=5*(x-1)**2',0.
vvod: DB 'Введите x: ',0
ans: DB 'ответ: ',0

SECTION .bss ;секция не иницированных данных
x: RESB 80 ;переменная размером 80 байт

SECTION .text ;код программы
GLOBAL _start ;начало программы
_start: ;точка входа в программу

mov eax,task ;адрес строки 'task' в 'eax'
call sprintf ;вызов подпрограммы печати
mov eax,vvod ;адрес строки 'vvod' в 'eax'
call sprintf ;вызов подпрограммы печати
mov ecx,x ;адрес переменной под вводимую строку
mov edx,80 ;длина вводимой строки
call sread ;вызов подпрограммы ввода

mov eax,x ;eax=x
call atoi ;вызов подпрограммы преобразования ASCII кода в число

dec eax ;вычитание 1
mov ebx,eax ;ebx=eax
mul ebx ; eax=x*x
mov ebx,5 ;ebx=5
mul ebx ; eax=x*5
mov ebx,eax ;ebx=eax

mov eax,ans ;адрес строки 'ans' в 'eax'
call sprintf ;вызов подпрограммы печати
mov eax,ebx ;eax=ebx
call sprintf ;вызов подпрограммы печати
call quit ;вызов подпрограммы завершения
```

Рис. 3.1: Текст программы

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab6-4.asm`, `ld -m elf_i386 -o lab6-4 lab6-4.o`, запуск: `./lab6-4`) (рис. 3.2)


```

oskashnikova@dk6n62:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
oskashnikova@dk6n62:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
oskashnikova@dk6n62:~/work/arch-pc/lab06$ ./lab6-4
f(x)=5*(x-1)**2
Введите x: 3
ответ: 20
oskashnikova@dk6n62:~/work/arch-pc/lab06$ ./lab6-4
f(x)=5*(x-1)**2
Введите x: 5
ответ: 80
oskashnikova@dk6n62:~/work/arch-pc/lab06$ █

```

Рис. 3.2: Создание и запуск исполняемого файла

Текст программы:

```

#include 'in_out.asm' ;подключение внешнего файла
SECTION .data ;секция инициированных данных
    task: DB 'f(x)=5*(x-1)**2',0
    vvod: DB 'Введите x: ',0
    ans: DB 'ответ: ',0
SECTION .bss ;секция не инициированных данных
    x: RESB 80 ;переменная размером 80 байт
SECTION .text ;код программы
GLOBAL _start ;начало программы
_start: ;точка входа в программу
    mov eax,task ;адрес строки 'task' в 'eax'
    call sprintf ;вызов подпрограммы печати
    mov eax,vvod ;адрес строки 'vvod' в 'eax'
    call sprintf ;вызов подпрограммы печати
    mov ecx,x ;адрес переменной под вводимую строку
    mov edx,80 ;длина вводимой строки
    call sread ;вызов подпрограммы ввода
    mov eax,x ;eax=x
    call atoi ;вызов подпрограммы преобразования ASCII кода в число
    dec eax ;вычитание 1
    mov ebx,eax ;ebx=eax

```

```
mul ebx ; eax=x*x
mov ebx,5 ;ebx=5
mul ebx ; eax=x*5
mov ebx,eax ;ebx=eax
mov eax, ans ;адрес строки 'ans' в 'eax'
call sprintf ;вызов подпрограммы печати
mov eax,ebx ;eax=ebx
call iprintLF ;вызов подпрограммы печати
call quit ;вызов подпрограммы завершения
```

4 Выводы

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