

# **Отчет по лабораторной работе № 7**

**Дисциплина: Архитектура данных**

Никулина Ксения Ильинична

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>5</b>	<b>Самостоятельная работа</b>	<b>15</b>
<b>6</b>	<b>Ответы на вопросы</b>	<b>18</b>
<b>7</b>	<b>Выводы</b>	<b>19</b>

# Список иллюстраций

4.1	Создание файла lab7-1.asm . . . . .	7
4.2	Текст . . . . .	7
4.3	Запуск файла . . . . .	8
4.4	Замена символов в тексте . . . . .	8
4.5	Запуск файла . . . . .	9
4.6	Создание файла lab7-2.asm . . . . .	9
4.7	Текст . . . . .	9
4.8	Запуск файла . . . . .	9
4.9	Замена символов . . . . .	10
4.10	Запуск файла . . . . .	10
4.11	Замена функции . . . . .	11
4.12	Запуск файла . . . . .	11
4.13	Создание файла lab7-3 . . . . .	12
4.14	Запуск файла . . . . .	12
4.15	Замена текста . . . . .	13
4.16	Запуск файла . . . . .	13
4.17	Создание файла . . . . .	13
4.18	Текст . . . . .	14
4.19	Запуск файла . . . . .	14
5.1	Создание файла . . . . .	15
5.2	Текст . . . . .	16
5.3	Ответ при $x = 1$ . . . . .	17
5.4	Ответ при $x = 9$ . . . . .	17

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

## 2 Задание

Освоить арифметические инструкции языка ассемблера NASM

### 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. Далее рассмотрены все существующие способы задания адреса хранения операндов способы адресации. Существует три основных способа адресации: • 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

Для выполнения лабораторных работ в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Это: • `iprint` – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число (`mov eax,`). • `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки. • `atoi` – функция преобразует `ascii`-код символа в целое число и запишет результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число (`mov eax,`).

## 4 Выполнение лабораторной работы

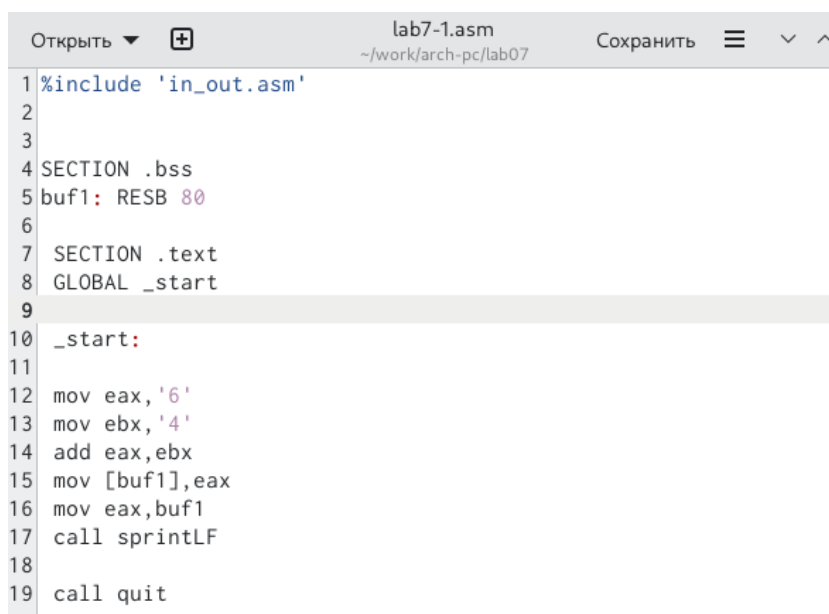
1. Создала каталог для программ лабораторной работы №7, перешла в него и создала файл lab7-1.asm (рис. 4.1)



```
kinikulina@dk8n72:~/work/arch-pc/lab07
kinikulina@dk8n72 ~ $ mkdir ~/work/arch-pc/lab07
kinikulina@dk8n72 ~ $ cd ~/work/arch-pc/lab07
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 4.1: Создание файла lab7-1.asm

2. Ввела в файл lab7-1.asm текст программы из листинга 7.1. (рис. 4.2)



```
lab7-1.asm
~/work/arch-pc/lab07
Сохранить

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

Рис. 4.2: Текст

3. Создала исполняемый файл и запустила его. (рис. 4.3)

```
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ./lab7-1
j
kinikulina@dk8n72 ~/work/arch-pc/lab07 $
```

Рис. 4.3: Запуск файла

4. Далее изменила текст программы и вместо символов записал в регистры числа. (рис. 4.4)

```
%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. 4.4: Замена символов в тексте

5. Создала исполняемый файл и запустила его. (рис. 4.5)



```
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ./lab7-1

kinikulina@dk8n72 ~/work/arch-pc/lab07 $
```

Рис. 4.5: Запуск файла

6. Создала файл lab7-2.asm (рис. 4.6)

```
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ touch lab7-2.asm
kinikulina@dk8n72 ~/work/arch-pc/lab07 $
```

Рис. 4.6: Создание файла lab7-2.asm

7. Ввела в него текст программы из листинга 7.2 (рис. 4.7)

```
Открыть + lab7-2.asm Сохранить
~/work/arch-pc/lab07
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7     mov eax, '6'
8     mov ebx, '4'
9     add eax, ebx
10    call iprintLF
11
12    call quit
```

Рис. 4.7: Текст

8. Создала исполняемый файл и запустила его. (рис. 4.8)

```
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ./lab7-2
106
```

Рис. 4.8: Запуск файла

9. Аналогично предыдущему примеру изменила символы на числа. (рис. 4.9)

```
kinikulina@dk8n72:~/work/arch-pc/lab07
GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru
%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: Замена символов

10. Создала исполняемый файл и запустила его. (рис. 4.10)

```
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ./lab7-2
10
kinikulina@dk8n72 ~/work/arch-pc/lab07 $
```

Рис. 4.10: Запуск файла

11. Заменяла функцию iprintLF на iprint. (рис. 4.11)

```
kinikulina@dk8n72:~/work/arch-pc/lab07
GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru/home/
#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: Замена функции

12. Создала исполняемый файл и запустила его. (рис. 4.12)

```
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ./lab7-2
kinikulina@dk8n72 ~/work/arch-pc/lab07 $
```

Рис. 4.12: Запуск файла

13. Создала файл lab7-3.asm в каталоге ~/work/arch-pc/lab07 (рис. 4.13)

```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,5 ; EAX=5
10 mov ebx,2 ; EBX=2
11 mul ebx ; EAX=EAX*EB
12 add eax,3 ; EAX=EAX+3
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,3 ; EBX=3
15 div ebx ; EAX=EAX/3, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершени

```

Рис. 4.13: Создание файла lab7-3

14. Создала исполняемый файл и запустила его. (рис. 4.14)

```

kinikulina@dk8n72 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ./lab7-3
Результат: 4
Остаток от деления: 1

```

Рис. 4.14: Запуск файла

15. Изменила текст программы для вычисления выражения  $\boxtimes(\boxtimes) = (4 \boxtimes 6 + 2)/5$  (рис. 4.15)

```
GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru/home/k/i/
#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*EB
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 завершени
```

Рис. 4.15: Замена текста

16. Создала исполняемый файл и запустила его. (рис. 4.16)

```
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ d -m elf_i386 -o lab7-3 lab7-3.o
bash: d: команда не найдена
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ ./lab7-3
Результат: 5
Остаток от деления: 1
```

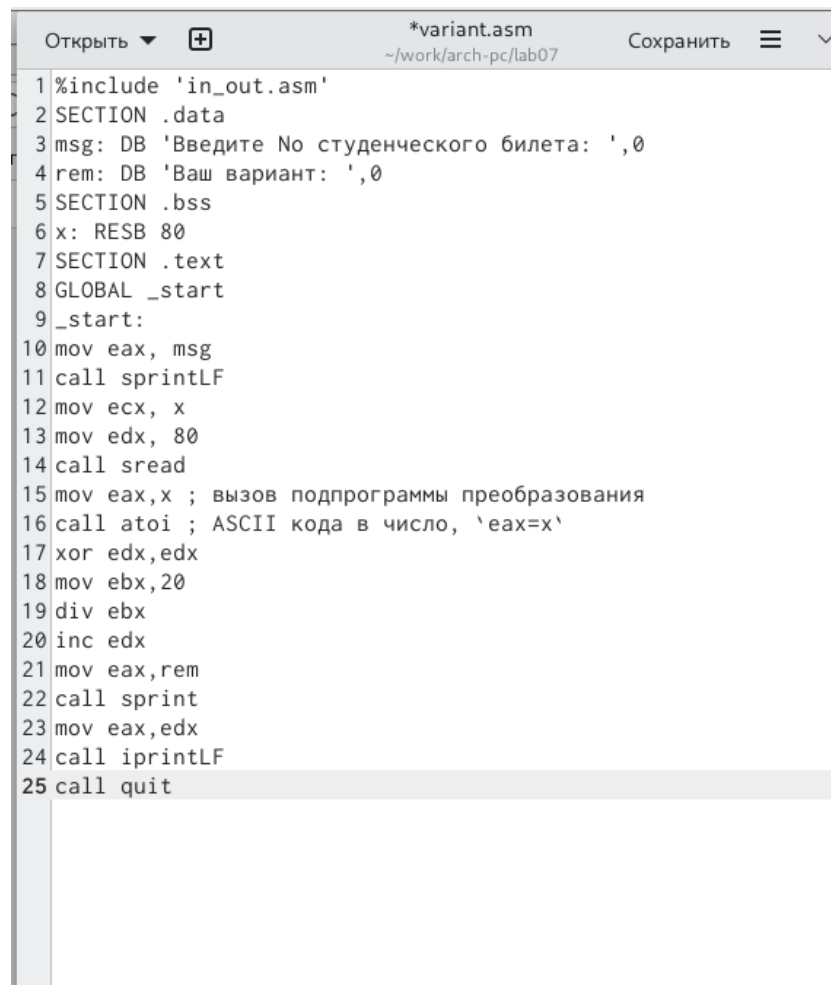
Рис. 4.16: Запуск файла

17. Создала файл variant.asm в каталоге ~/work/arch-pc/lab07 (рис. 4.17)

```
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ touch variant.asm
```

Рис. 4.17: Создание файла

18. Ввела в файле текст (рис. 4.18)

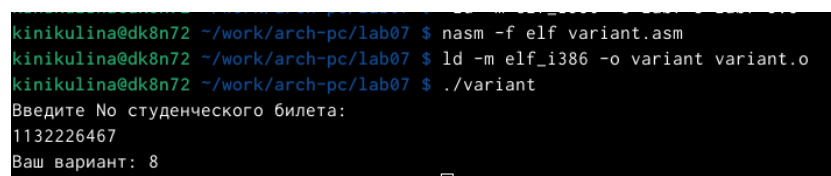


The screenshot shows a text editor window titled '\*variant.asm' with a menu bar containing 'Открыть', a plus icon, the filename, the path '~/.work/arch-pc/lab07', 'Сохранить', and a dropdown arrow. The code is as follows:

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите No студенческого билета: ',0
4 rem: DB 'Ваш вариант: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax,x ; вызов подпрограммы преобразования
16 call atoi ; ASCII кода в число, 'eax=x'
17 xor edx,edx
18 mov ebx,20
19 div ebx
20 inc edx
21 mov eax,rem
22 call sprintf
23 mov eax,edx
24 call iprintLF
25 call quit
```

Рис. 4.18: Текст

19. Создала исполняемый файл и запустила его. (рис. 4.19)



The screenshot shows a terminal window with the following commands and output:

```
kinikulina@dk8n72 ~/.work/arch-pc/lab07 $ nasm -f elf variant.asm
kinikulina@dk8n72 ~/.work/arch-pc/lab07 $ ld -m elf_i386 -o variant variant.o
kinikulina@dk8n72 ~/.work/arch-pc/lab07 $ ./variant
Введите No студенческого билета:
1132226467
Ваш вариант: 8
```

Рис. 4.19: Запуск файла

## 5 Самостоятельная работа

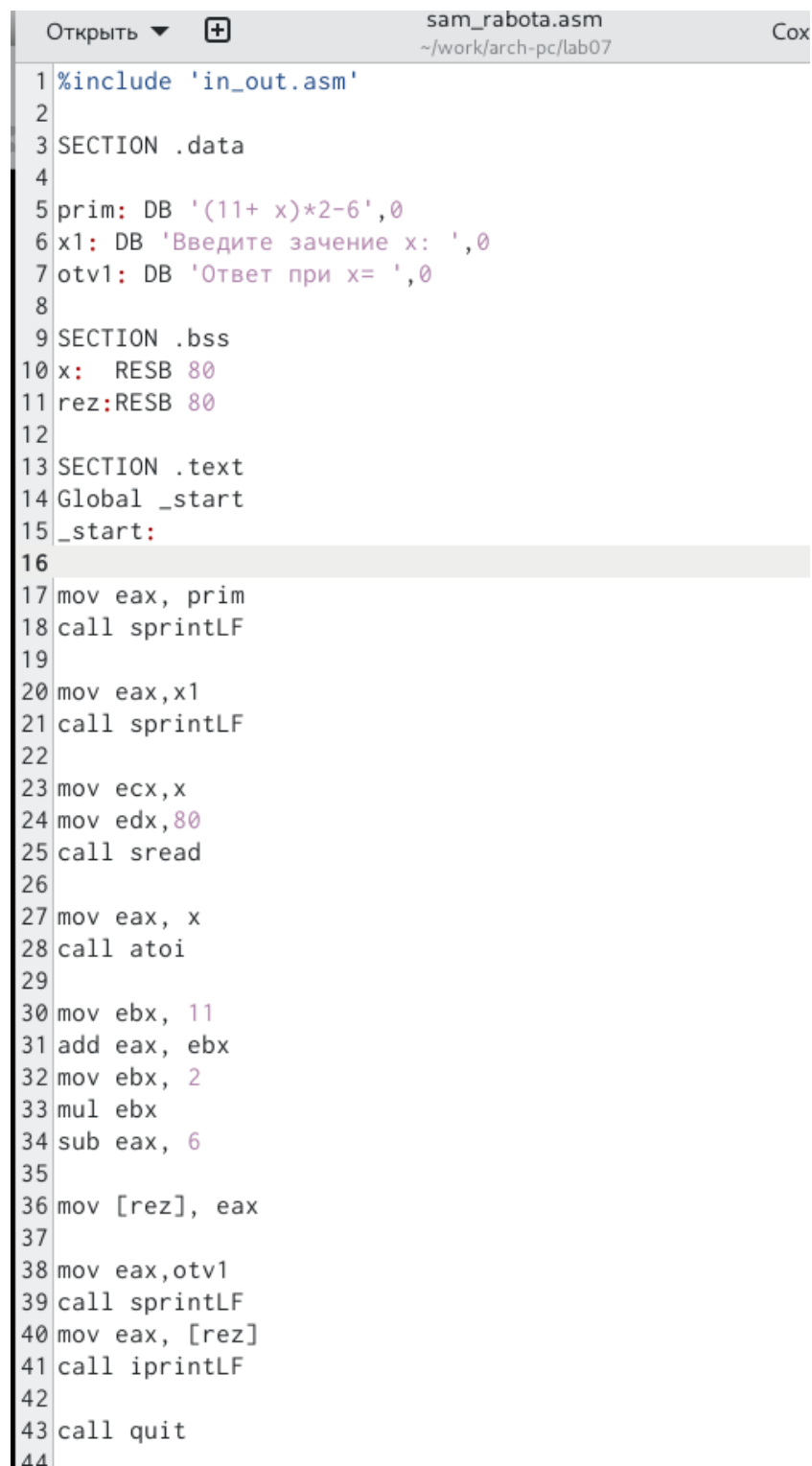
1. Создала файл для самостоятельной работы (рис. 5.1)



```
kinikulina@dk8n72 ~/work/arch-pc/lab07 $ touch sam_rabota.asm
```

Рис. 5.1: Создание файла

2. Ввела программу, которая будет решать уравнение  $(11 + \boxed{x}) \cdot 2 - 6$  (рис. 5.2)



```

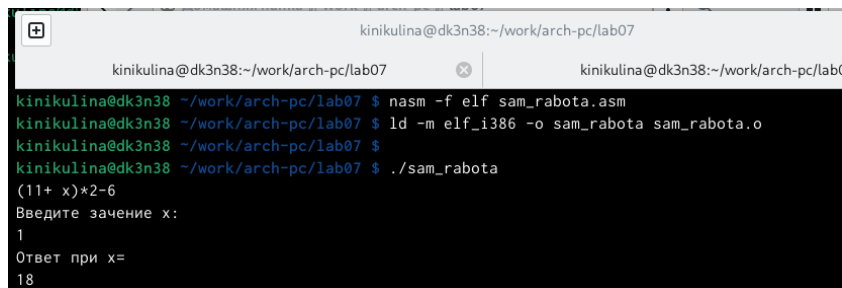
1 %include 'in_out.asm'
2
3 SECTION .data
4
5 prim: DB '(11+ x)*2-6',0
6 x1: DB 'Введите значение x: ',0
7 otv1: DB 'Ответ при x= ',0
8
9 SECTION .bss
10 x: RESB 80
11 rez: RESB 80
12
13 SECTION .text
14 Global _start
15 _start:
16
17 mov eax, prim
18 call sprintLF
19
20 mov eax, x1
21 call sprintLF
22
23 mov ecx, x
24 mov edx, 80
25 call sread
26
27 mov eax, x
28 call atoi
29
30 mov ebx, 11
31 add eax, ebx
32 mov ebx, 2
33 mul ebx
34 sub eax, 6
35
36 mov [rez], eax
37
38 mov eax, otv1
39 call sprintLF
40 mov eax, [rez]
41 call iprintLF
42
43 call quit
44

```

Рис. 5.2: Текст

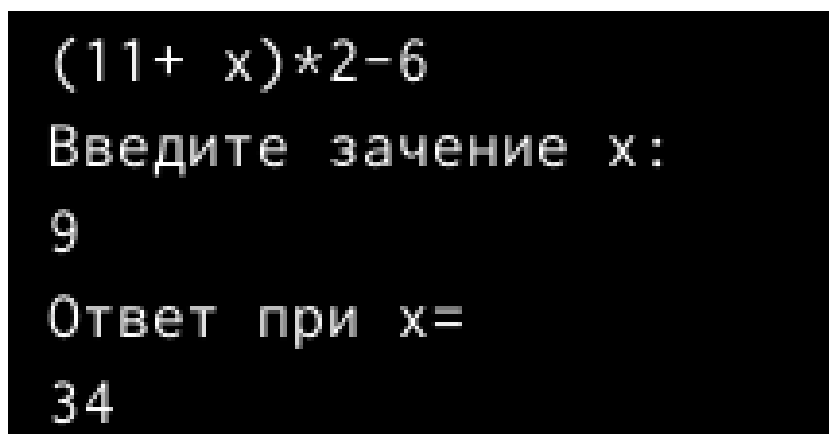
3. Создала исполняемый файл и запустила его. (рис. 5.3, рис. 5.4)





```
kinikulina@dk3n38:~/work/arch-pc/lab07
kinikulina@dk3n38:~/work/arch-pc/lab07
kinikulina@dk3n38 ~/work/arch-pc/lab07 $ nasm -f elf sam_rabota.asm
kinikulina@dk3n38 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o sam_rabota sam_rabota.o
kinikulina@dk3n38 ~/work/arch-pc/lab07 $
kinikulina@dk3n38 ~/work/arch-pc/lab07 $ ./sam_rabota
(11+ x)*2-6
Введите значение x:
1
Ответ при x=
18
```

Рис. 5.3: Ответ при  $x = 1$



```
(11+ x)*2-6
Введите значение x:
9
Ответ при x=
34
```

Рис. 5.4: Ответ при  $x = 9$

## 6 Ответы на вопросы

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? `mov eax,msg call sprintLF`
2. Для чего используются следующие инструкции? `asm mov ecx, x mov edx, 80 call sread` Эти инструкции используются для ввода переменной X с клавиатуры и сохранения введенных данных.
3. Для чего используется инструкция “`call atoi`”? Эта инструкция используется для преобразования Кода переменной ASCII в число.
4. Какие строки листинга 7.4 отвечают за вычисления варианта? `mov ebx,20 div ebx inc edx`
5. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”? В регистре `ebx`.
6. Для чего используется инструкция “`inc edx`”? Для увеличения значения `edx` на 1.
7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений? `mov eax,edx call iprintLF`

## 7 Выводы

Я освоила арифметические инструкции языка ассемблера NASM