

Лабораторная работа №6

Дисциплина: Архитектура компьютеров

Лаптев Тимофей Сергеевич

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	Ответы на вопросы:	13
4.3	Задание для самостоятельной работы	14
5	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	Создание каталога и файла	9
4.2	Запуск файла	9
4.3	Запуск файла	10
4.4	Запуск файла	10
4.5	Запуск файла	11
4.6	Запуск файла	11
4.7	Запуск файла	12
4.8	Запуск файла	12
4.9	Запуск файла	13
4.10	Запуск файла	15

Список таблиц

1 Цель работы

Целью данной лабораторной работы является освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Символьные и численные данные в NASM.
2. Выполнение арифметических операций в NASM.
3. Задание 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. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа адресации: • регистровая адресация; • непосредственная адресация; • адресация памяти. Схема команды целочисленного сложения `add` (от англ. *addition* - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда `add` работает как с числами со знаком, так и без знака. Допустимые сочетания операндов для команды `add` аналогичны сочетаниям операндов для команды `mov`. Так, например, команда `add eax,ebx` прибавит значение из регистра `eax` к значению из регистра `ebx` и запишет результат в регистр `eax`. Команда целочисленного вычитания `sub` (от англ. *subtraction* – вычитание) работает аналогично команде `add`. Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: `inc` (от англ. *increment*) и `dec` (от англ. *decrement*), которые увеличивают и уменьшают на 1 свой операнд. Операндом может быть регистр или ячейка памяти любого размера. Команды инкремента и декремента выгодны тем, что они занимают меньше места, чем соответствующие команды сложения и вычитания. Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют

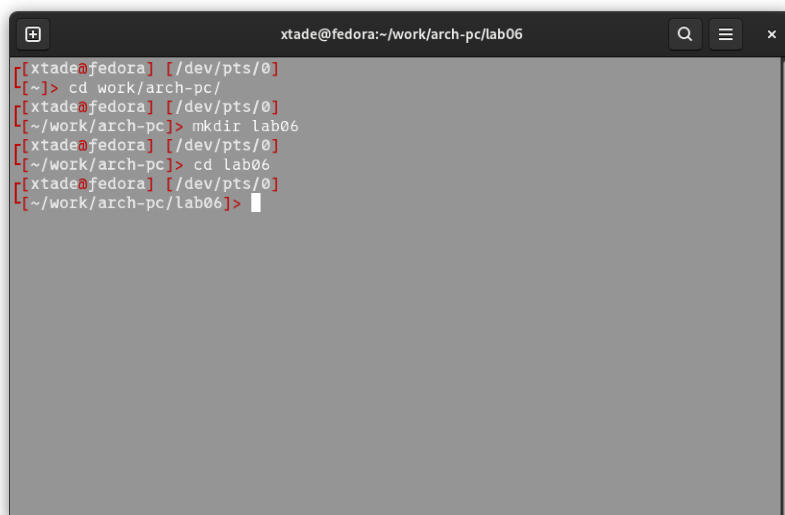
различные команды. Для беззнакового умножения используется команда `mul`, для знакового умножения используется команда `imul`. Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. 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 записывает результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число (`mov eax,`).

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

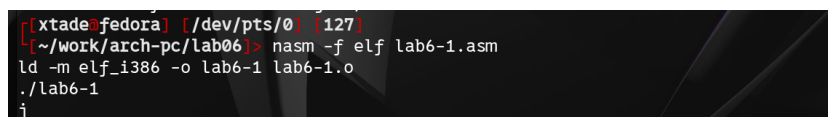
Сначала создаю каталог для программ лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm (рис. 4.1).



```
xtade@fedora: ~/work/arch-pc/lab06
[xtade@fedora] [/dev/pts/0]
[~]> cd work/arch-pc/
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc]> mkdir lab06
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc]> cd lab06
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]>
```

Рис. 4.1: Создание каталога и файла

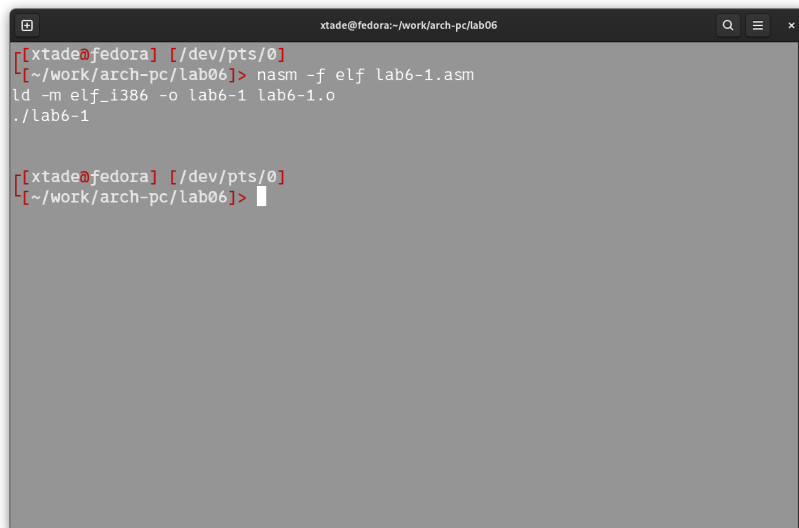
Ввожу в файл lab6-1.asm текст программы из листинга, далее создаю исполняемый файл и запускаю его (рис. 4.2).



```
xtade@fedora [ /dev/pts/0 ] 127
[~/work/arch-pc/lab06]> nasm -f elf lab6-1.asm
ld -m elf_i386 -o lab6-1 lab6-1.o
./lab6-1
j
```

Рис. 4.2: Запуск файла

Затем изменяю текст программы и вместо символов, записываю в регистры числа: 'б', '4' заменяю на 4, 6. Создаю исполняемый файл и запускаю его (рис. 4.3). Это символ перевода строки, он не отображается при выводе на экран.

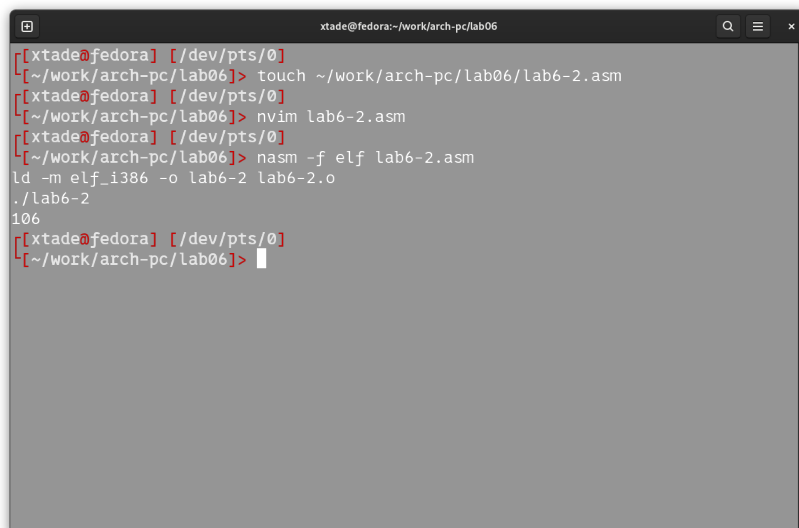


```
xtade@fedora: ~/work/arch-pc/lab06
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> nasm -f elf lab6-1.asm
ld -m elf_i386 -o lab6-1 lab6-1.o
./lab6-1

[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]>
```

Рис. 4.3: Запуск файла

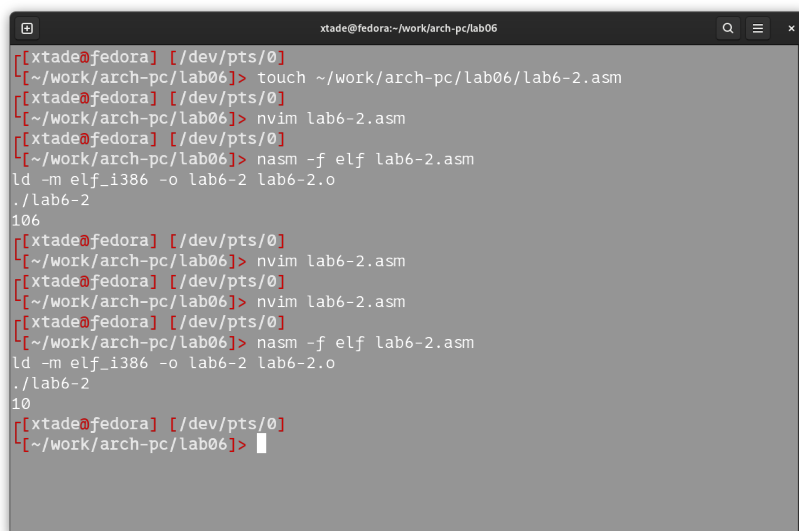
Далее создаю файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввожу в него текст программы из листинга. Создаю исполняемый файл и запускаю его (рис. 4.4).



```
xtade@fedora: ~/work/arch-pc/lab06
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> touch ~/work/arch-pc/lab06/lab6-2.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> nvim lab6-2.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
./lab6-2
106
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]>
```

Рис. 4.4: Запуск файла

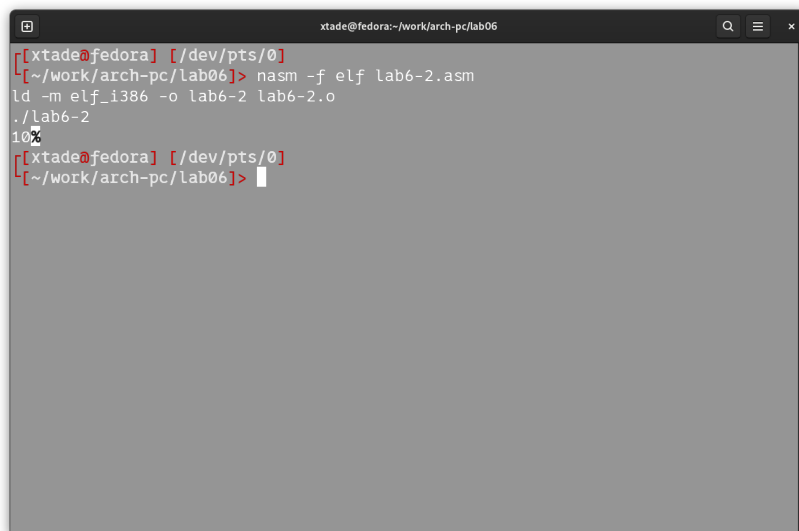
Аналогично предыдущему примеру изменяю символы на числа. Создаю исполняемый файл и запускаю его (рис. 4.5). Программа складывает числа 6 и 4, поэтому вывод 10.



```
xtade@fedora: ~/work/arch-pc/lab06
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06] touch ~/work/arch-pc/lab06/lab6-2.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06] nvim lab6-2.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06] nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
./lab6-2
106
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06] nvim lab6-2.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06] nvim lab6-2.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06] nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
./lab6-2
10
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06] 
```

Рис. 4.5: Запуск файла

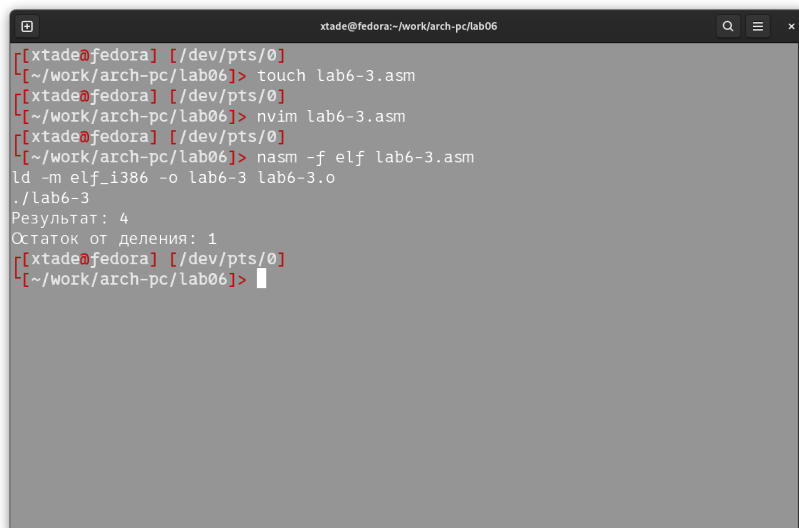
Заменяю функцию `iprintLF` на `iprint`, создаю исполняемый файл и запускаю его (рис. 4.6). Вывод функций отличается тем, что `iprint` не добавляет в выводе символ переноса строки.



```
xtade@fedora: ~/work/arch-pc/lab06
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06] nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
./lab6-2
10%
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06] 
```

Рис. 4.6: Запуск файла

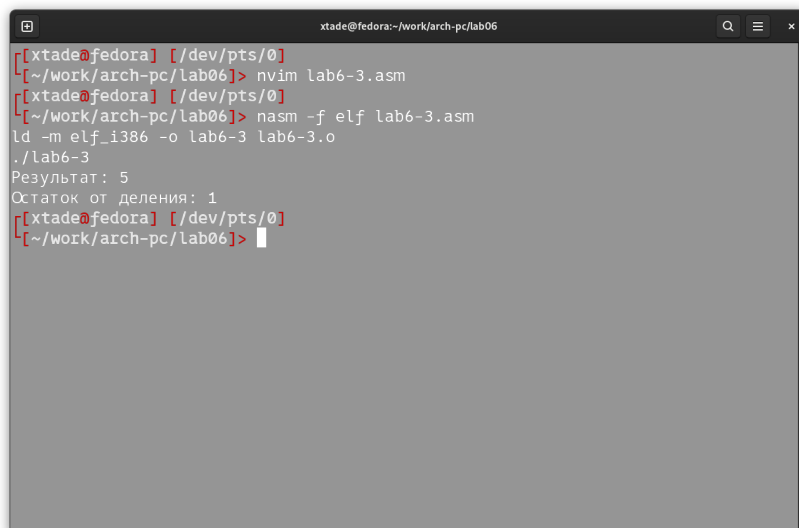
Затем создаю файл lab6-3.asm в каталоге ~/work/arch-pc/lab06. Ввожу текст программы из листинга 6.3 в lab6-3.asm. Создаю исполняемый файл и запускаю его (рис. 4.7).



```
xtade@fedora: ~/work/arch-pc/lab06
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> touch lab6-3.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> nvim lab6-3.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> nasm -f elf lab6-3.asm
ld -m elf_i386 -o lab6-3 lab6-3.o
./lab6-3
Результат: 4
Остаток от деления: 1
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]>
```

Рис. 4.7: Запуск файла

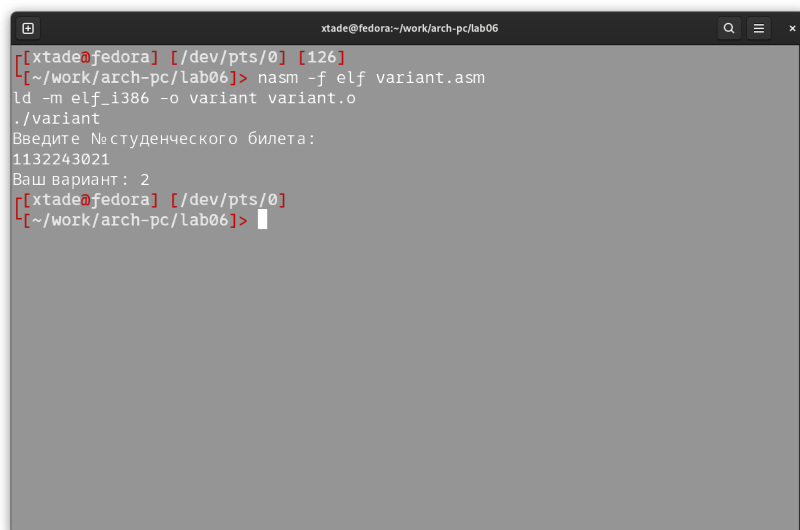
Изменяю текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$. Создаю исполняемый файл и проверяю его работу (рис. 4.8).



```
xtade@fedora: ~/work/arch-pc/lab06
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> nvim lab6-3.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> nasm -f elf lab6-3.asm
ld -m elf_i386 -o lab6-3 lab6-3.o
./lab6-3
Результат: 5
Остаток от деления: 1
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]>
```

Рис. 4.8: Запуск файла

Далее делаю программу вычисления варианта задания для самостоятельной работы по номеру студенческого билета. Создаю файл `variant.asm` в каталоге `~/work/arch-pc/lab06`. Ввожу текст программы из листинга 6.4 в файл `variant.asm`. Создаю исполняемый файл и запускаю его (рис. ??).



```
xtade@fedora:~/work/arch-pc/lab06
[xtade@fedora] [/dev/pts/0] [126]
[~/work/arch-pc/lab06]> nasm -f elf variant.asm
ld -m elf_i386 -o variant variant.o
./variant
Введите № студенческого билета:
1132243021
Ваш вариант: 2
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]>
```

Рис. 4.9: Запуск файла

4.2 Ответы на вопросы:

1. За вывод на экран сообщения 'Ваш вариант:' отвечают следующие строки:
`mov eax,rem call sprint`
2. `mov ecx, x` используется, чтобы положить адрес вводимой строки в регистр, `mov edx, 80` используется для записи в регистр длины вводимой строки, `call sread` вызывает подпрограмму из внешнего файла, чтобы вводить сообщения с клавиатуры.
3. Используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр.
4. За вычисление варианта отвечают следующие строки: `xor edx,edx mov ebx,20 div ebx inc edx`
5. В регистр `edx`.

6. `inc edx` используется для увеличения значения регистра `edx` на 1.
7. За вывод на экран результатов вычислений отвечают следующие строки:
`mov eax,edx call iprintfLF`

4.3 Задание для самостоятельной работы

Сначала создаю файл `laptevvariant2.asm`. (рис. 4.9) Далее ввожу в файл текст программы для вычисления значения выражения $(8x + 6) \cdot 10$ (вариант 2) (рис. ??).

```
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> touch laptevvariant2.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> nvim laptevvariant2.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]>
```

```
20 %include 'in_out.asm' ; подключение внешнего файла
19 SECTION .data
18 msg: DB 'Введите значение переменной x: ',0
17 div: DB 'Результат: ',0
16 SECTION .bss
15 x: RESB 80
14 SECTION .text
13 GLOBAL _start
12 _start:
11 ; ---- Ввод x
10 mov eax, msg
9 call sprintf
8 mov ecx, x
7 mov edx, 80
6 call sread
5 ; ---- Вывод значения
4 mov eax, msg
3 mov ecx, x ; вызов подпрограммы преобразования
2 call atoi ; ASCII кода в число, 'eaxx'
1 mov ebx, 12 ; запись значения 12 в регистр ebx
21 mul ebx, EAX ; EAX = EAX * EBX
20 add eax, 6 ; EAX = EAX + 6
19 mov ebx, 5 ; EAX = EAX * EBX
18 mul ebx
17 mov edi, eax ; запись результата вычисления в 'edi'
16 ; ---- Вывод результата на экран
15 mov eax, div ; вызов подпрограммы печати
14 call sprintf ; сообщения 'Результат: '
13 mov eax, edi ; вызов подпрограммы печати значения
12 call iprintfLF ; из 'edi' в виде символов
11 call quit ; вызов подпрограммы завершения
```

Создаю и запускаю исполняемый файл. Проверил несколько значений, программа работает исправно (рис. 4.10).

```
xtade@fedora:~/work/arch-pc/lab06
[~/work/arch-pc/lab06]> nasm -f elf laptevvariant2.asm
ld -m elf_i386 -o laptevvariant2 laptevvariant2.o
./laptevvariant2
Введите значение переменной x: 3
Результат: 195
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> ./laptevvariant2
Введите значение переменной x: 4
Результат: 255
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> ./laptevvariant2
Введите значение переменной x: 7
Результат: 435
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> ./laptevvariant2
Введите значение переменной x: 10
Результат: 615
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> ./laptevvariant2
Введите значение переменной x: 1
Результат: 75
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab06]> █
```

Рис. 4.10: Запуск файла

5 Выводы

В ходе данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

Список литературы