Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютеров

Зайцева Ульяна Владимировна

Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

# 2 Задание

1. Символьные и численные данные в NASM.
2. Выполнение арифметических операций в NASM.
3. Ответы на вопросы по листингу 6.4
4. 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 бытьданные 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.

Схема команды целочисленного сложения add (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда.

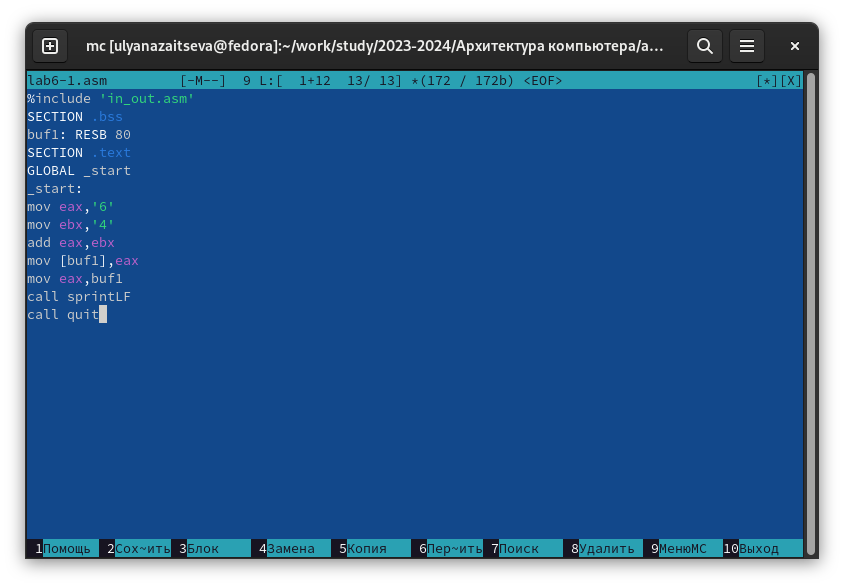
Команда целочисленного вычитания sub (от англ. subtraction – вычитание) работает аналогично команде add

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. Расширенная таблица ASCII состоит из двух частей. Первая (символы с кодами 0-127) является универсальной, а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться.

# 4 Выполнение лабораторной работы

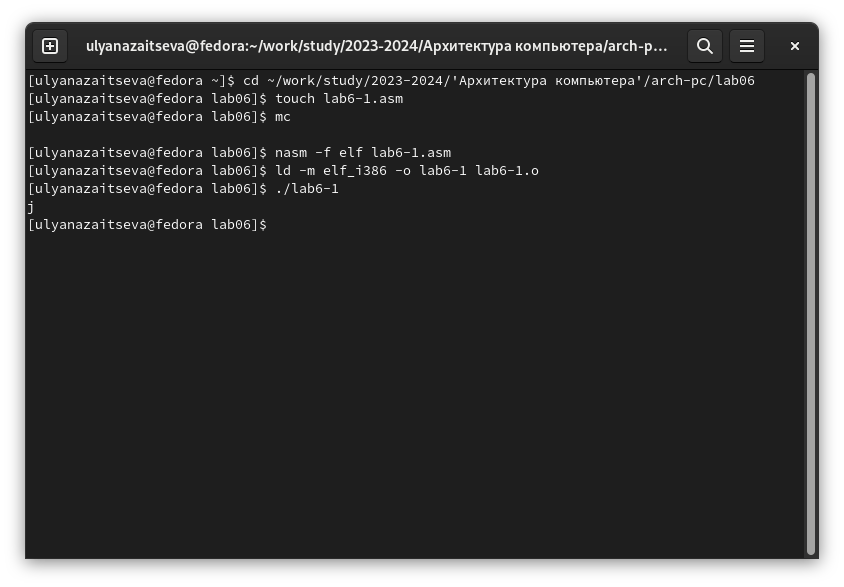
1. Символьные и численные данные в NASM.

Ввожу в файл lab6-1.asm текст программы из листинга 6.1.(рис. ??)



Первый листинг

Создаю исполняемый файл и запускаю его.(рис. ??)



Запуск программы

Далее меняю текст программы и вместо символов запишу числа. Исправляю текст программы:

Меняю строки

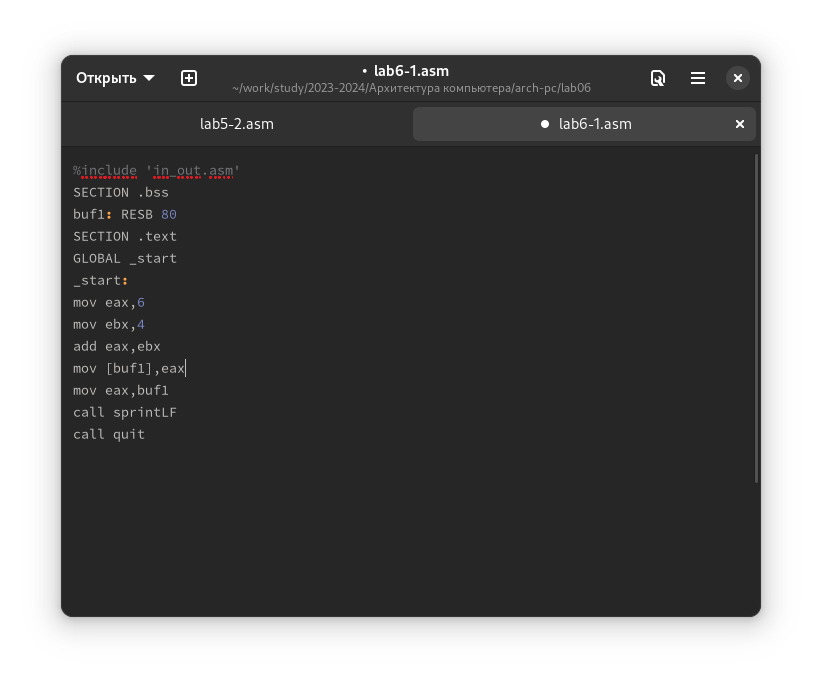
mov eax,‘6’

mov ebx,‘4’

на строки

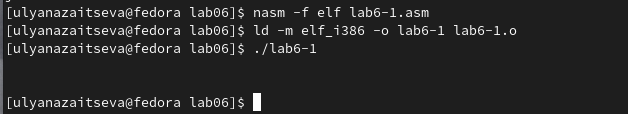
mov eax,6

mov ebx,4 (рис. ??)



Изменённая программа

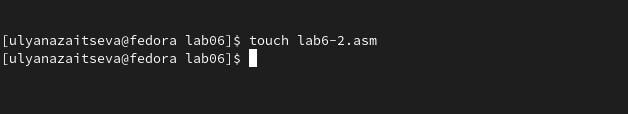
Создаю исполняемый файл и запускаю его.(рис. ??)



Выполнение программы

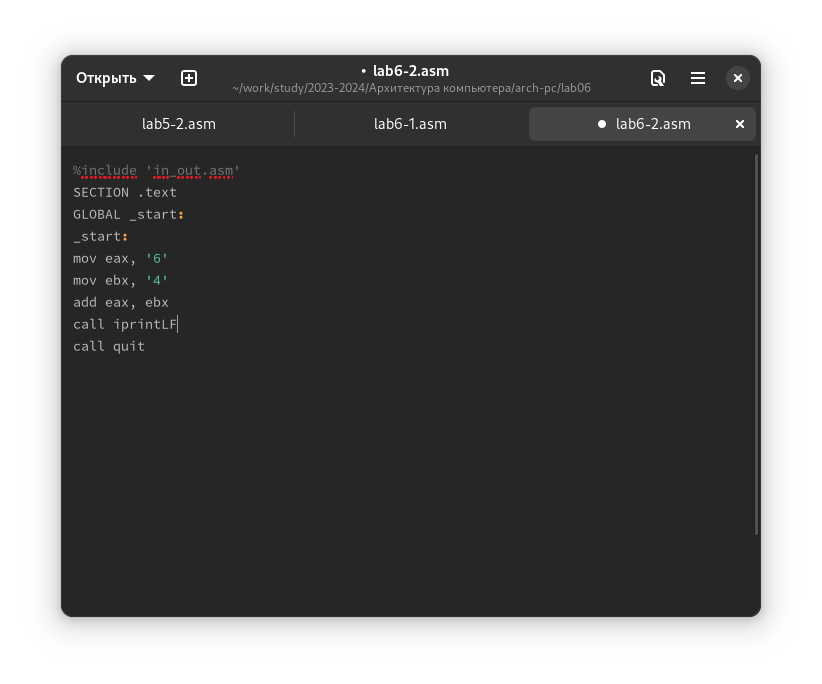
Этому коду (10) соответствует символ ‘LF, ’, который перемещает курсор на следующую строку. Cимвол при выводе не отображается.

Создаю файл lab6-2.asm в нужном каталоге(рис. ??)



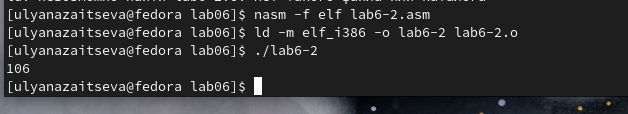
Выполнение программы

Ввожу в него текст программы из листинга 6.2(рис. ??).



Второй листинг

Создаю исполняемый файл и запускаю его.(рис. ??)



Вывод программы

В этой программе также заменяю строки

mov eax,‘6’

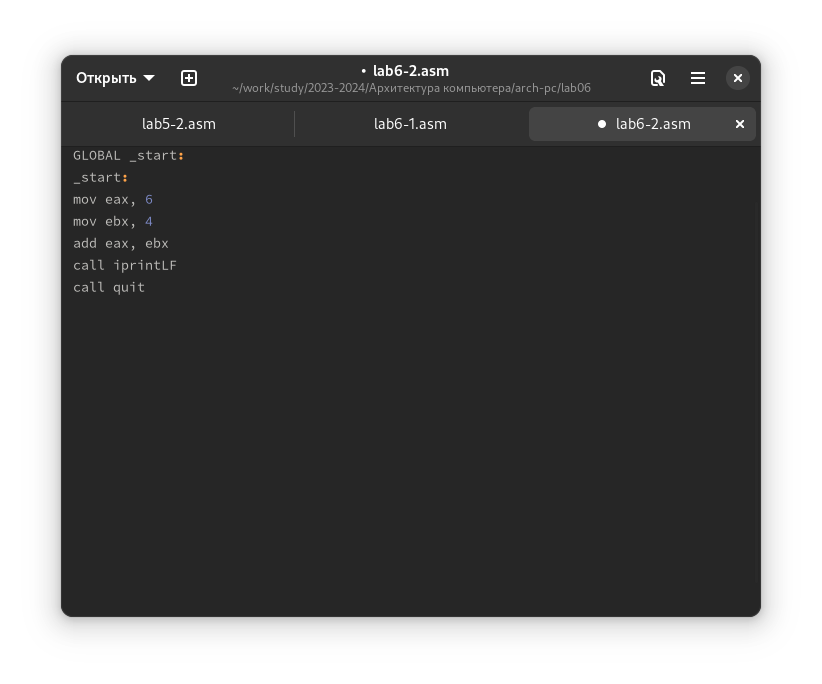
mov ebx,‘4’

на строки

mov eax,6

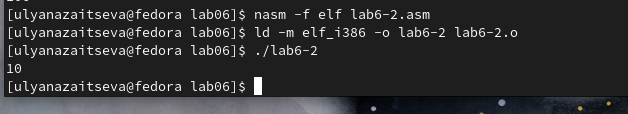
mov ebx,4

(рис. ??).



Изменяю программу

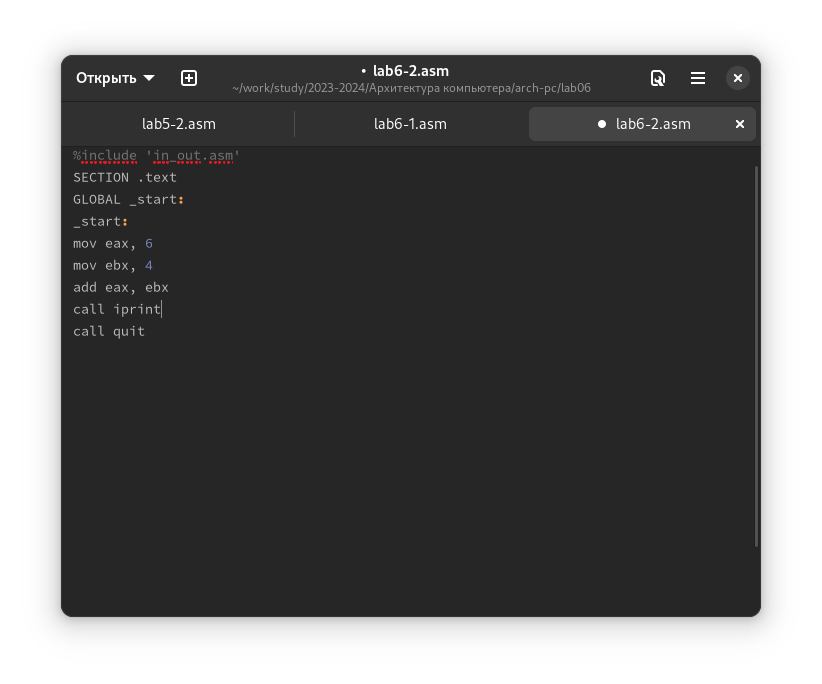
Создаю исполняемый файл и запускаю его.(рис. ??)



Проверка работы программы

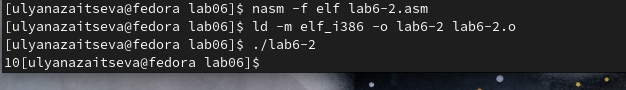
В выводе получаю число 10.

Заменяю функцию iprintLF на iprint.(рис. ??)



Замена в программе

Создаю исполняемый файл и запускаю его.(рис. ??)

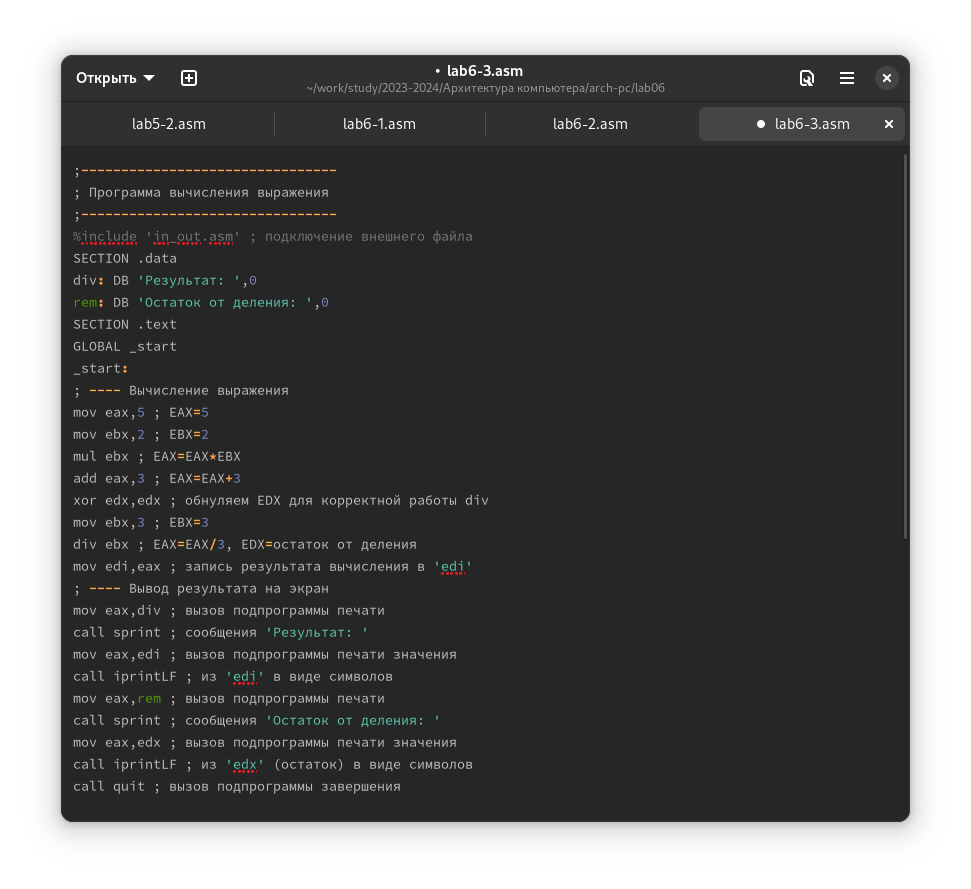


Проверка

Вывод функций iprintLF и iprint отличается тем, что при использовании iprintLF выполняется перенос на следующую строку после вывода, а при использовании iprint этого не происходит.

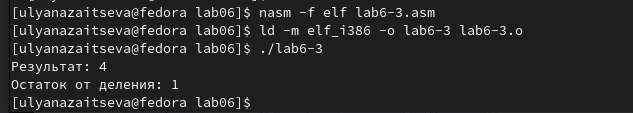
1. Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm и ввожу в него текст из листинга 6.3.(рис. ??)



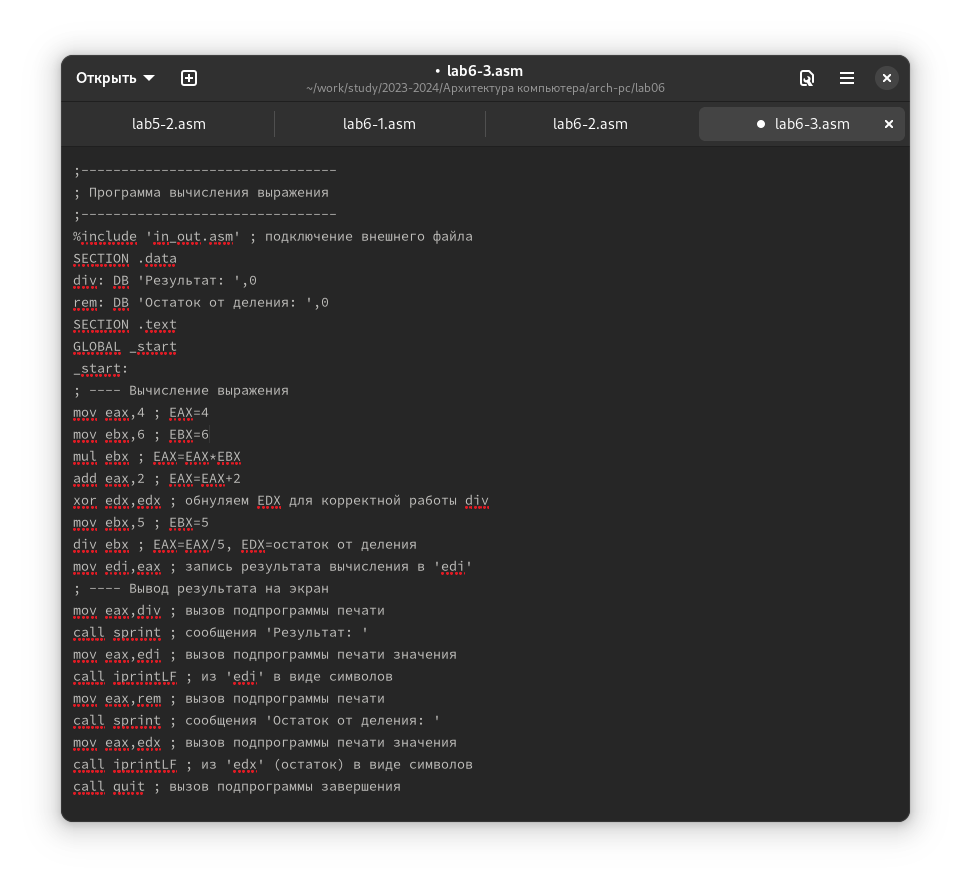
Третий листинг

Создаю исполняемый файл и запускаю его.(рис. ??)



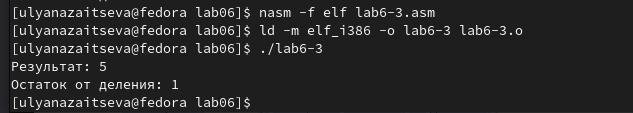
Третий листинг

Изменяю текст программы для вычисления выражения f(x) = (4 \* 6 + 2)/5, делая замену чисел в регистрах. (рис. ??)



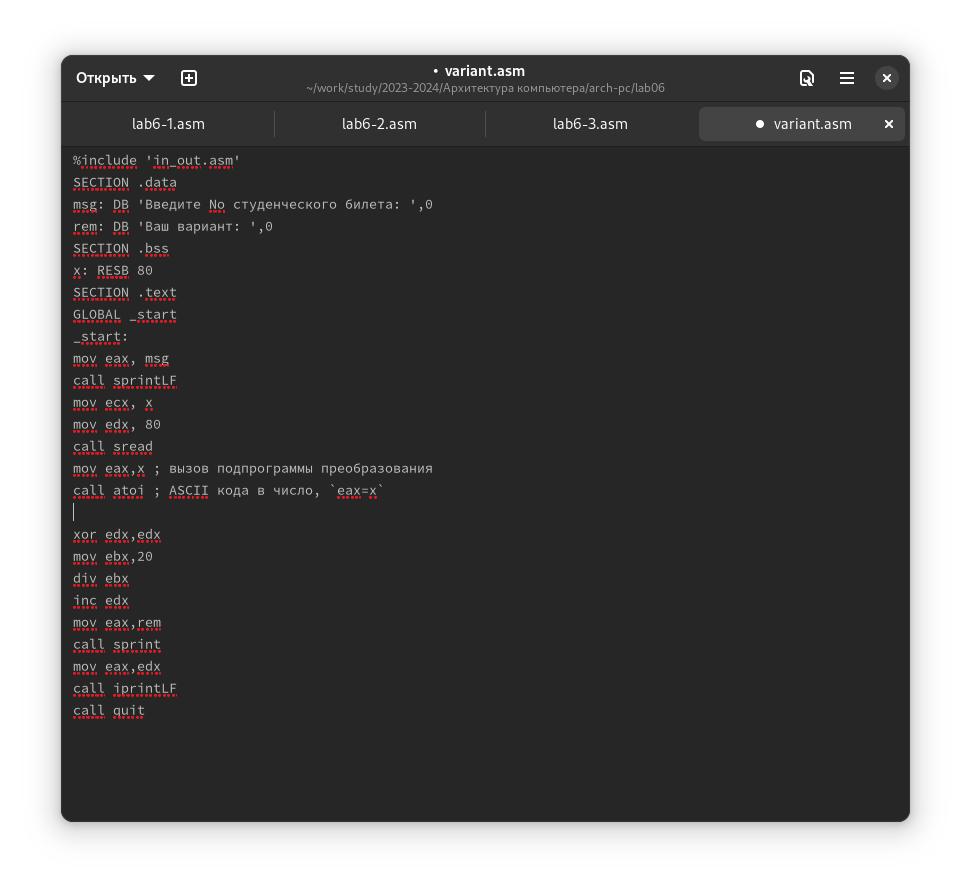
Изменяю текст программы

Создаю исполняемый файл и запускаю его.(рис. ??)



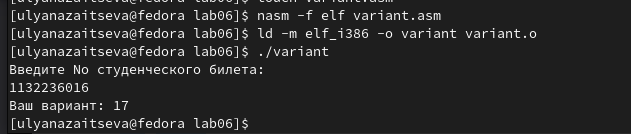
Проверяю работу программы

Создаю файл variant.asm. Ввожу текст программы из листинга 6.4, создаю исполняемый файл и запускаю его.(рис. ??)



Четвертый листинг

Проверяю результат работы программы, вычислив номер варианта.(рис. ??)



Вычисление моего варианта по номеру студенческого

1. Ответы на вопросы по листингу 6.4
2. За вывод сообщения “Ваш вариант” отвечают строки кода:

mov eax,rem

call sprint

1. mov ecx, x - закладывает адрес вводимой строки x в регистр.

mov edx, 80 - запись в регистр edx длины вводимой строки.

call sread - вызов подпрограммы из внешнего файла, обеспечивает ввод сообщения с клавиатуры.

1. ‘call atoi’ - вызов подпрограммы из внешнего файла, преобразующей ascii-код символа в целое число и записывает результат в регистр eax.
2. За вычисления варианта отвечают строки:

xor edx,edx

mov ebx,20

div ebx

inc edx

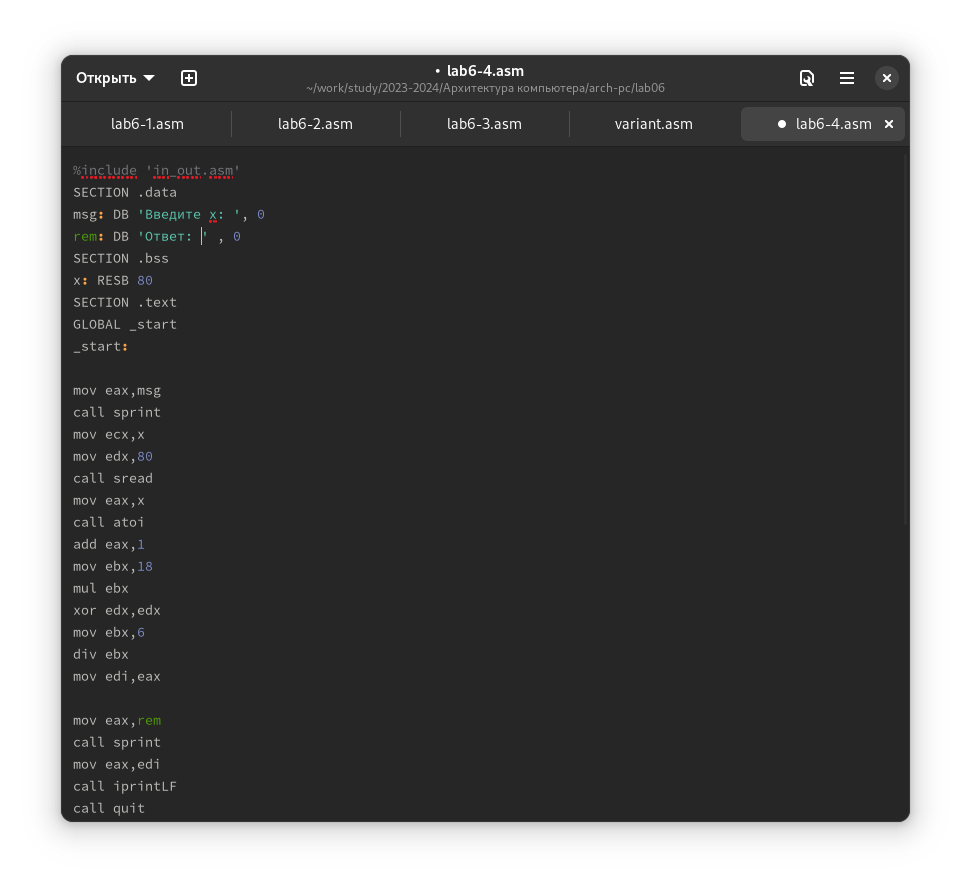
1. При выполнении инструкции div ebx остаток от деления записывается в регистр edx.
2. Инструкция “inc edx” увеличивает значение регистра edx на 1.
3. За вывод на экран результатов вычислений отвечают строки:

mov eax,edx

call iprintLF

1. Задание для самостоятельной работы

Мой варианта- 17, поэтому мне нужно написать программу для вычисления выражения 18(х + 1)/6 и проверить ее работу для значений х1 = 3 и х2 = 1.(рис. ??)



Программа для вычисления выражения

Код:

%include ‘in\_out.asm’

SECTION .data

msg: DB ‘Введите x:’, 0

rem: DB ‘Ответ:’ , 0

SECTION .bss

x: RESB 80

SECTION .text

GLOBAL \_start

\_start:

mov eax,msg

call sprint

mov ecx,x

mov edx,80

call sread

mov eax,x

call atoi

add eax,1

mov ebx,18

mul ebx

xor edx,edx

mov ebx,6

div ebx

mov edi,eax

mov eax,rem

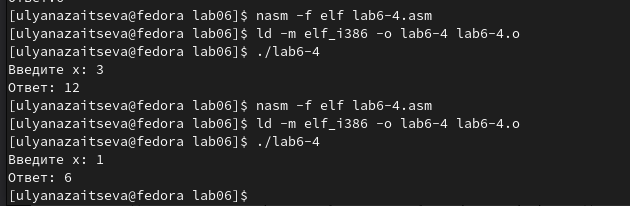
call sprint

mov eax,edi

call iprintLF

call quit

Создаю исполняемый файл и проверяю его работу.(рис. ??)



Проверка работы со значениями х1, х2

# 5 Выводы

С помощью данной лабораторной работы я освоила арифметические инструкции языка ассемблер NASM