Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

София Андреевна Кудякова

Содержание

# 1 Цель работы

Цель данной лабораторной работы - научиться пользоваться арифметическими инструкциями языка ассемблера NASM.

# 2 Задания

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

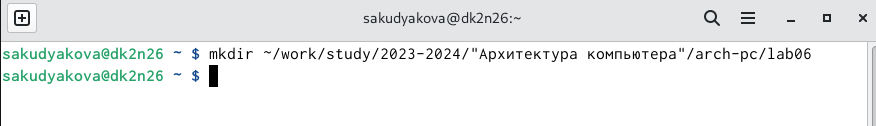
# 3 Теоретическое введение

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

# 4 Выполнение лабораторной работы

## 4.1 Символьные и численные данные в NASM

Ввожу команду mkdir, с помощью которой создаю директорию, в которой буду создавать файлы . (рис. ??).



Создание директории

С помощью команды touch создаю файл lab6-1.asm. (рис. ??).

Создание файла

Создание файла

С помощью cp копирую файл in\_out.asm в текущий каталог. (рис. ??).

Копирование файла

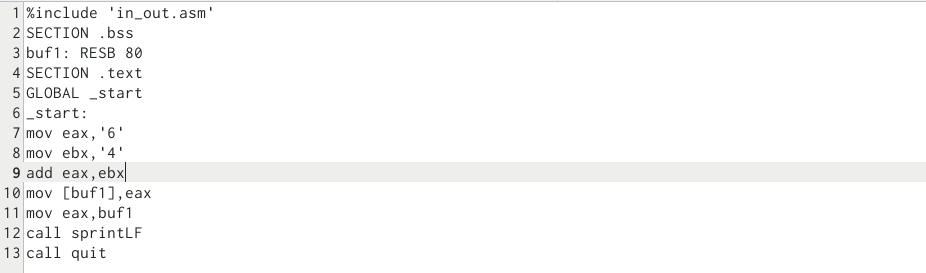
Копирование файла

Проверяю, корректность выполненных действий с помощью команды ls. (рис. ??).

Проверка действий

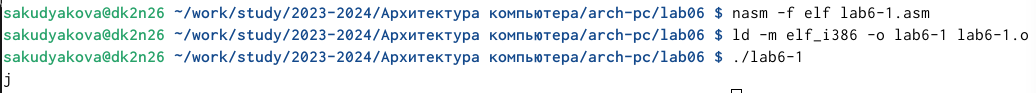
Проверка действий

Открываю созданный файл в редакторе и вставляю в него программу вывода значения регистра eax. (рис. ??).



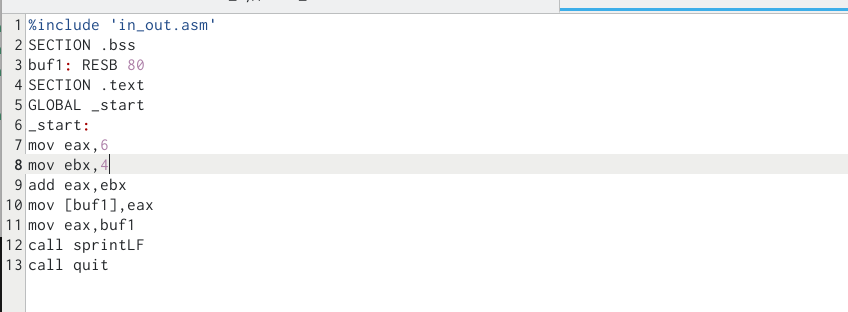
Редактирование файла

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



Запуск программы файла

Далее изменяю текст программы и вместо символов, записываю в регистры цифры 6 и 4. (рис. ??).



Редактирование файла

Создаю исполняемый файл и запускаю его. В данном случае выводится символ с кодом 10. Это символ перевода строки, этот символ не отображается при выводе на экран. (рис. ??).



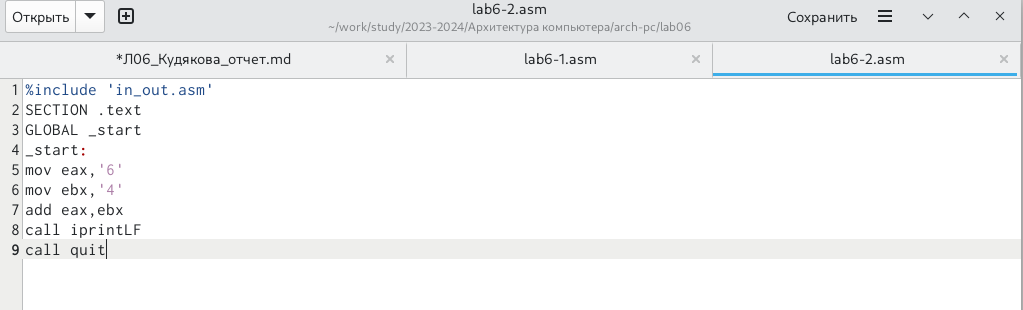
Исполнение файла

Создаю новый файл lab6-2.asm в каталоге ~/work/arch-pc/lab06. (рис. ??).

Создание файла

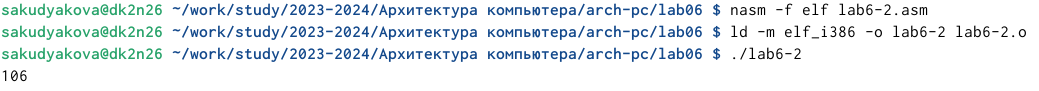
Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax. (рис. ??).



Ввод программы в файл

Создаю исполняемый файл и запускаю его. В результате работы программы получается число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”. (рис. ??).



Исполнение файла

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



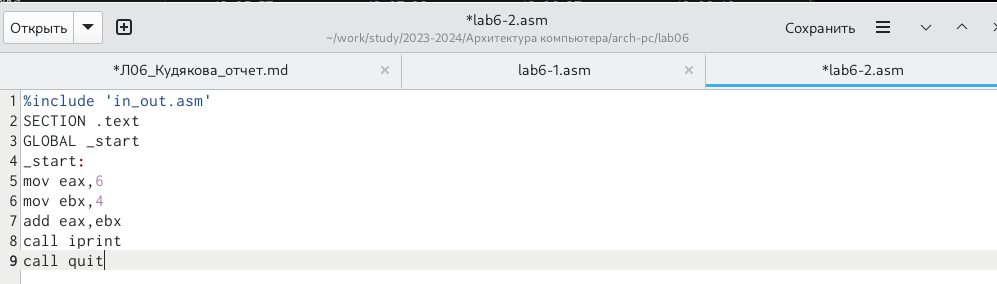
Редактирование файла

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



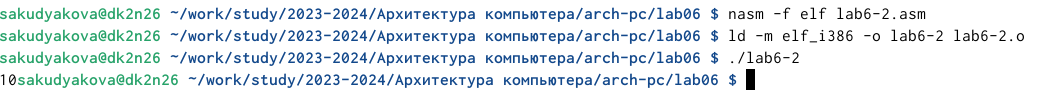
Исполнение файла

Далее заменяю функцию iprintLF на iprint . (рис. ??).



Редактирование файла

Создаю исполняемый файл и запускаю его. В результате работы программы можно сделать вывод: вывод функций iprintLF и iprint различается тем, что iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF. (рис. ??).



Исполнение файла

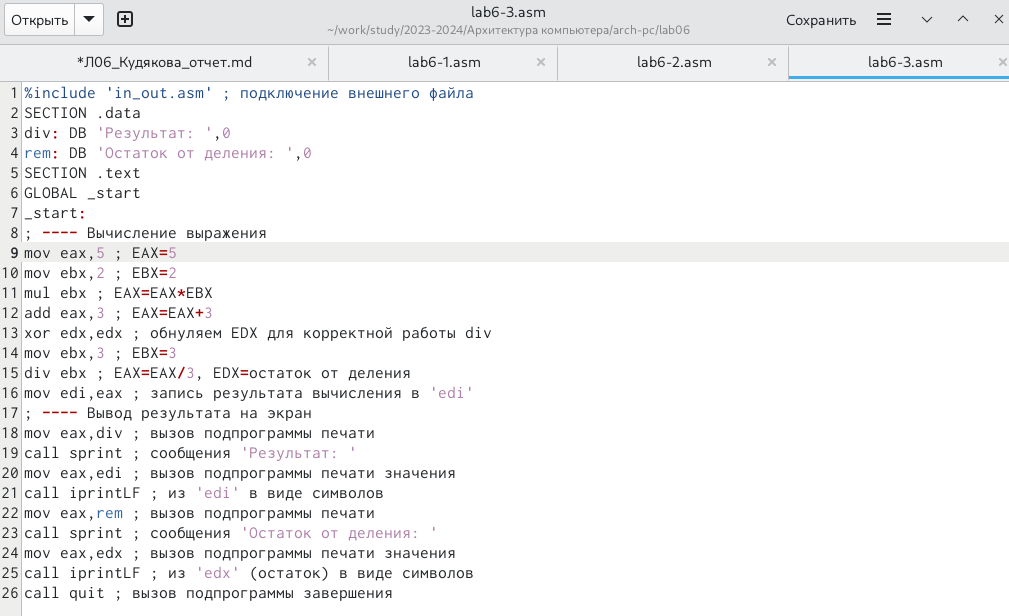
## 4.2 Выполнение арифметических действий в NASM

С помощью touch создаю файл lab6-3.asm. (рис. ??).

Создание файла

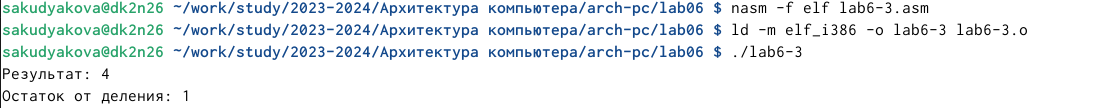
Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения f(x) = (5 \* 2 + 3)/3 (рис. ??).



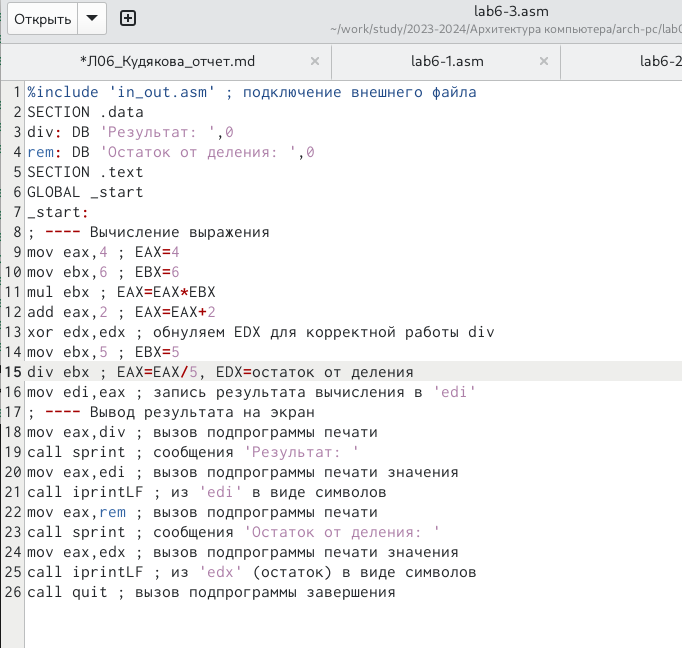
Редактирование файла

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



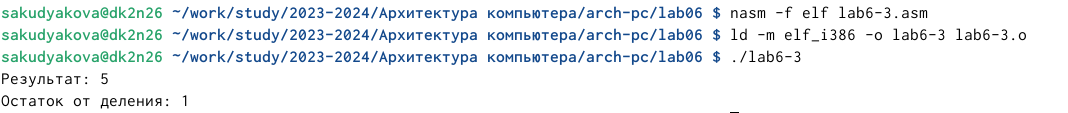
Исполнение файла

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5. (рис. ??).



Редактирование файла

Создаю исполняемый файл и запускаю его. Программа рабоает верно, так как я самостоятельно посчитала значение выражения, которое, в свою очередь, совпало с выводом программы. (рис. ??).



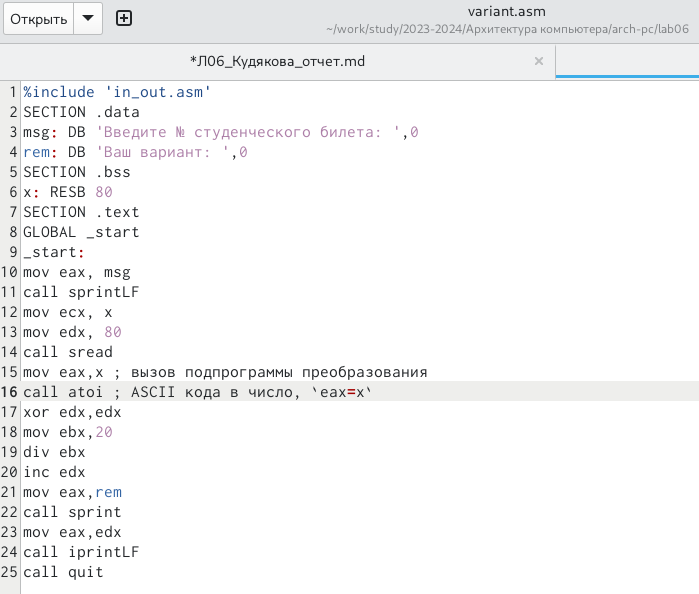
Запуск исполняемого файла

Создаю файл variant.asm. (рис. ??).

Создание файла

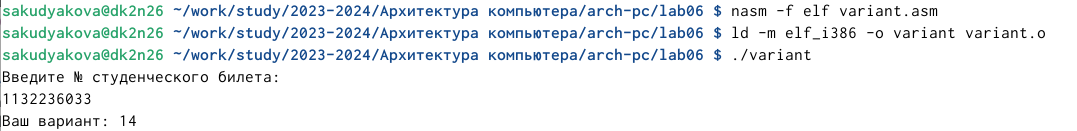
Создание файла

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



Ввод текста программы в файл

Создаю исполняемый файл и запускаю его. Ввожу номер своего студенческого билета, программа вчислила, что мой вариант - 14. (рис. ??).



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

1. Инструкция mov ecx, x используется, чтобы положить адрес вводимой стро- ки 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
2. call atoi используется для вызова подпрограммы из внешнего файла, кото- рая преобразует ascii-код символа в целое число и записывает результат в регистр eax
3. 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

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

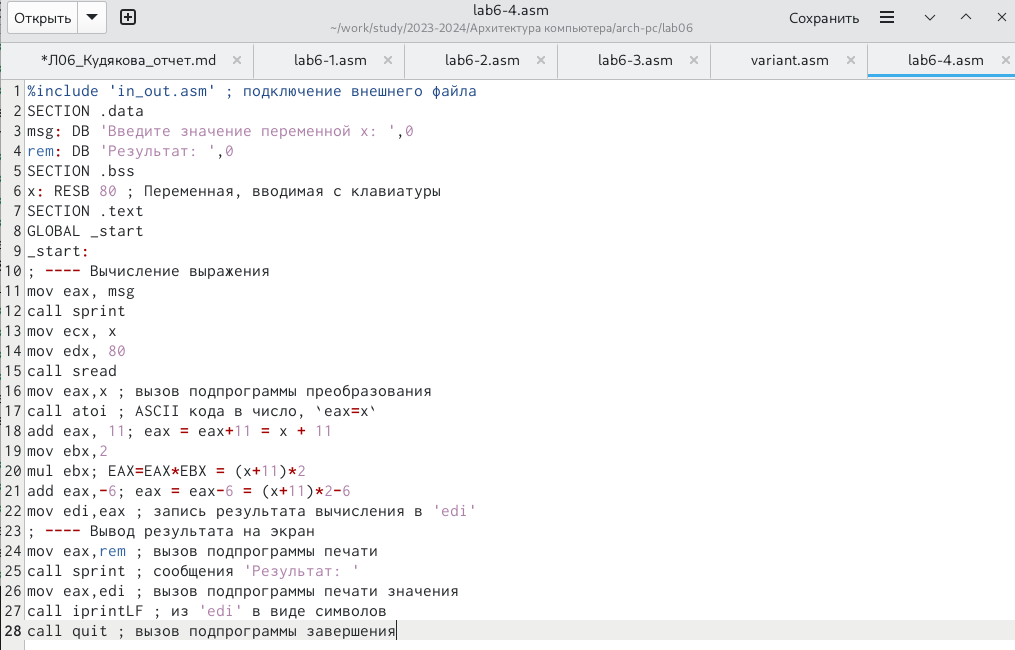
## 4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm. (рис. ??).

Создание файла

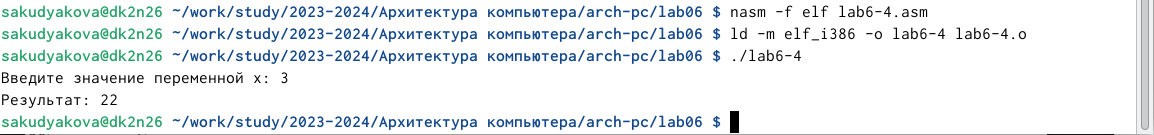
Создание файла

Из данных выражений я выбрала выражение №8 - (11 + x) \* 2 - 6. Ввожу в файл текст программы для вычисления значения выражения (рис. ??).



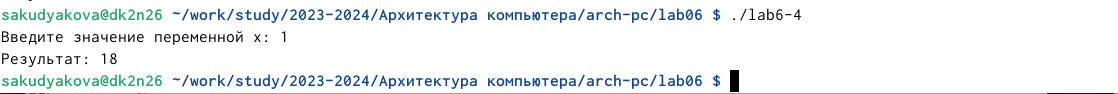
Ввод программы

Создаю исполняемый файл и запускаю его. Если задать переменной х значение 3, то вывод - 22. (рис. ??).



Исполнение файла

Для проверки работы программы ввожу другое значение х. (рис. 4.27). Программа отработала верно.



Исполнение файла

**Листинг 4.1. Программа для вычисления значения выражения (11 + x) \* 2− 6.**

%include 'in\_out.asm' ; подключение внешнего файла  
SECTION .data  
msg: DB 'Введите значение переменной x: ',0  
rem: DB 'Результат: ',0  
SECTION .bss  
x: RESB 80 ; Переменная, вводимая с клавиатуры   
SECTION .text  
GLOBAL \_start  
\_start:  
; ---- Вычисление выражения  
mov eax, msg  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax,x ; вызов подпрограммы преобразования  
call atoi ; ASCII кода в число, `eax=x`  
add eax, 11; eax = eax+11 = x + 11  
mov ebx,2  
mul ebx; EAX=EAX\*EBX = (x+11)\*2  
add eax,-6; eax = eax-6 = (x+11)\*2-6  
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 iprintLF ; из 'edi' в виде символов  
call quit ; вызов подпрограммы завершения

# 5 Выводы

В ходе данной лабораторной работы я научилась пользоваться арифметическими инструкциями языка ассемблера NASM.

# Список литературы

[Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089086/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%966.%20%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5%20%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8%20%D0%B2%20NASM..pdfu)