Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютера

Учаева Алёна Сергеевна

Содержание

Список иллюстраций

# 1 Цель работы

Цель данной лабораторной работы освоить арифметические инструкции языка ассемблера NASM.

# 2 Задание

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 опе- ранда 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 ко- манде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символи- ческое 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). По- этому, 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 дан- ные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.

# 4 Выполнение лабораторной работы

## 4.1 Символьные и численные данные в NASM

С помощью команды mkdir создаю каталог для программ лабораторной работы №6, перехожу в него и создаю файл lab6-1.asm (рис. 1).

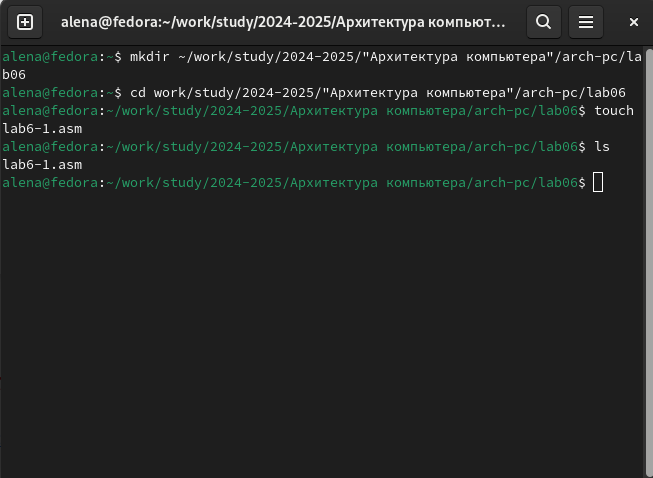


Рис. 1: Создание файла

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. 2).

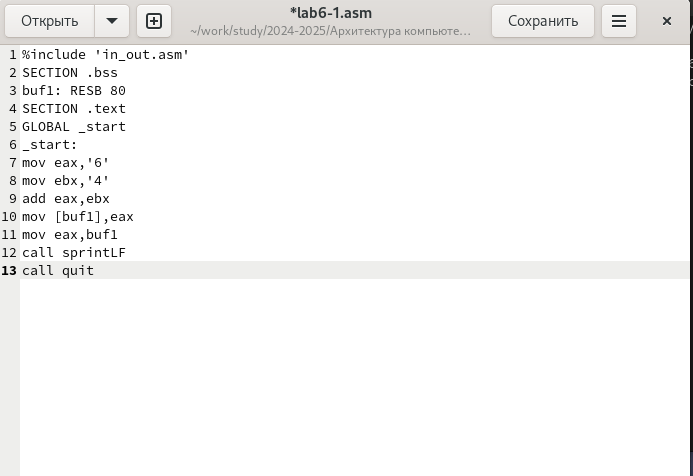


Рис. 2: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 3).

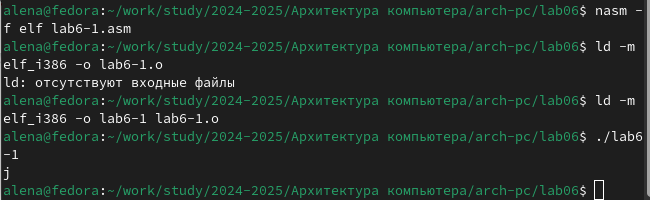


Рис. 3: Запуск исполняемого файла

Далее изменяю текст программы, вместо символов записываю в регистры числа (рис. 4).

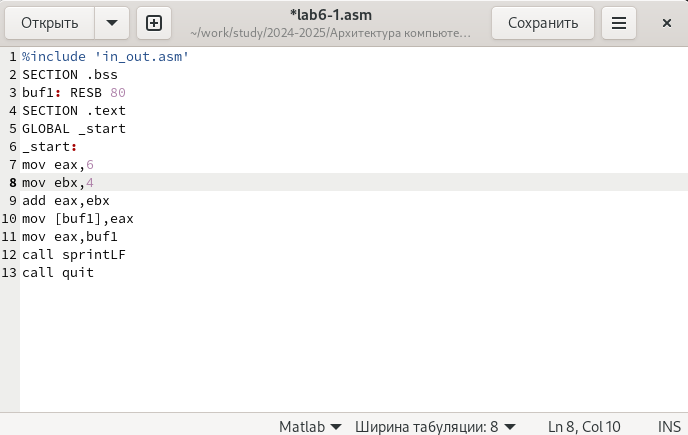


Рис. 4: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 5). Вывелся симыол с кодом 10,это символ перевода строки,этот символ не отображается при выводе на экран.

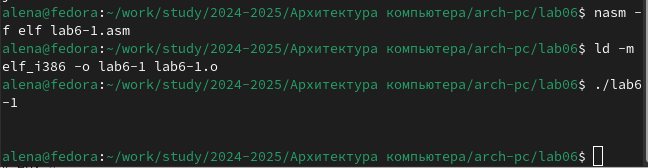


Рис. 5: Запуск исполняемого файла

С помощью команды touch создаю файл lab6-2.asm и ввожу в него текст программы (рис. 6).



Рис. 6: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 7).

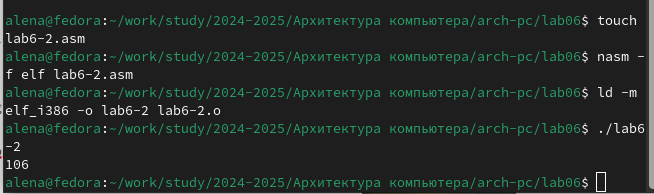


Рис. 7: Запуск исполняемого файла

Далее изменяю символы на числа (рис. 8).

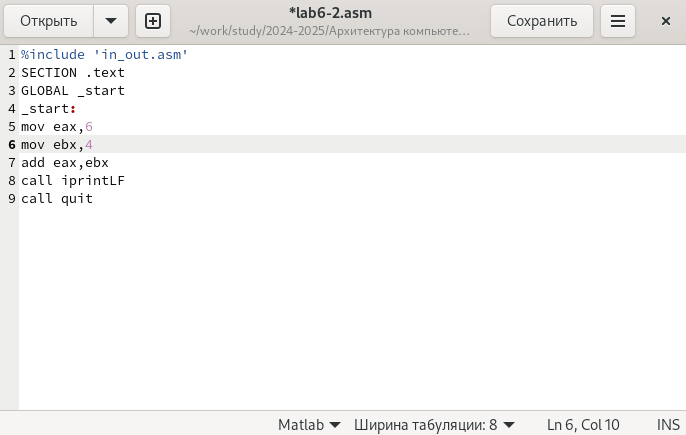


Рис. 8: Редактирование файла

Создаю и запусааю исполняемый файл (рис. 9). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому выводит 10.

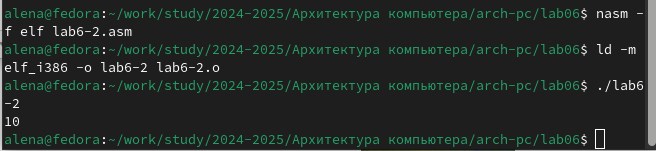


Рис. 9: Запуск исполняемого файла

Заменяю функцию iprintLF на Iprint (рис. 10).

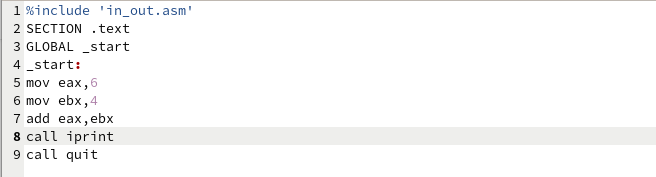


Рис. 10: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 11).Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

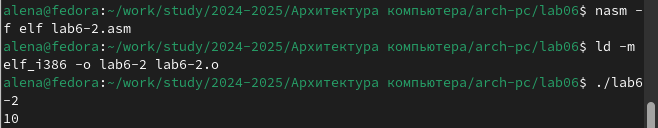


Рис. 11: Запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3 с помощью команды touch (рис. 12).

Рис. 12: Создание файла

Рис. 12: Создание файла

Ввожу текст программы для вычисления выражения 𝑓(𝑥)=(5\*2+3)/3 в исполняемый файл (рис. 13).

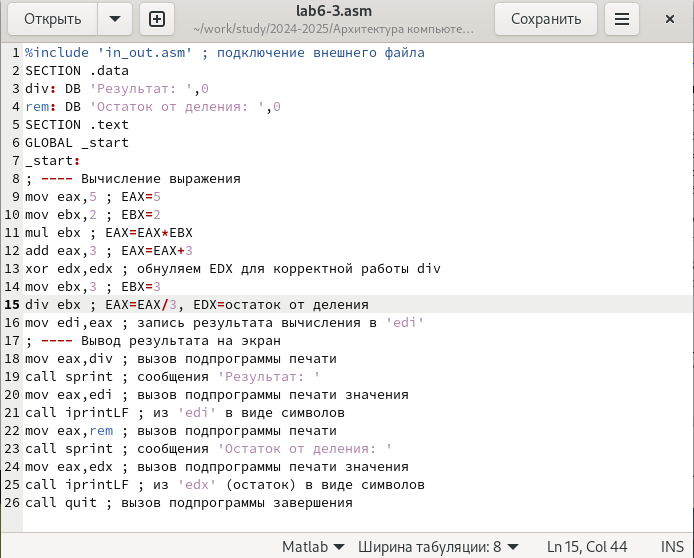


Рис. 13: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 14).

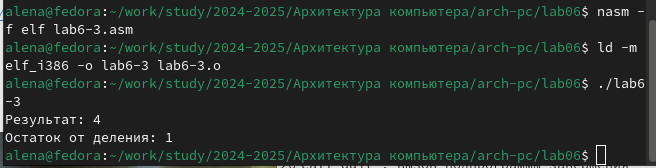


Рис. 14: Запуск исполняемого файла

Далее изменяю текст программы для вычисления выражения 𝑓(𝑥)=(4\*6+2)/5 (рис. 15).

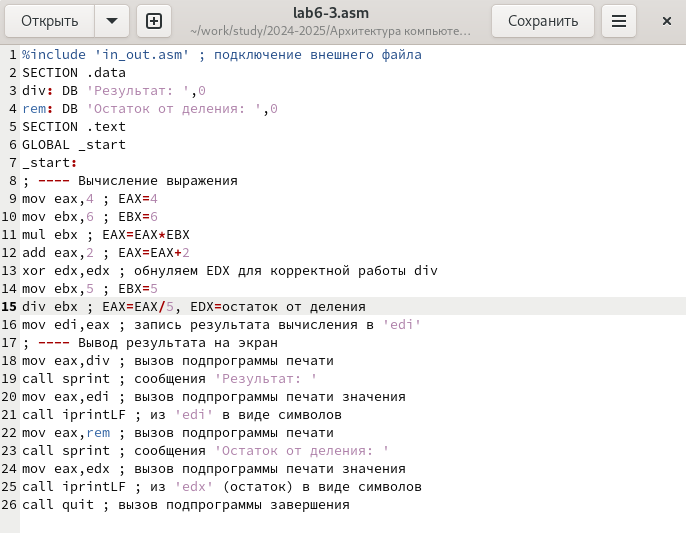


Рис. 15: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 16).

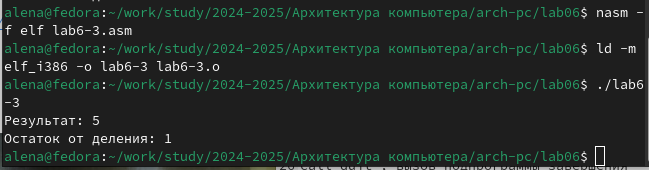


Рис. 16: Запуск исполняемого файла

Создаю файл variant.asm (рис. 17).

Рис. 17: Создание файла

Рис. 17: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 18).

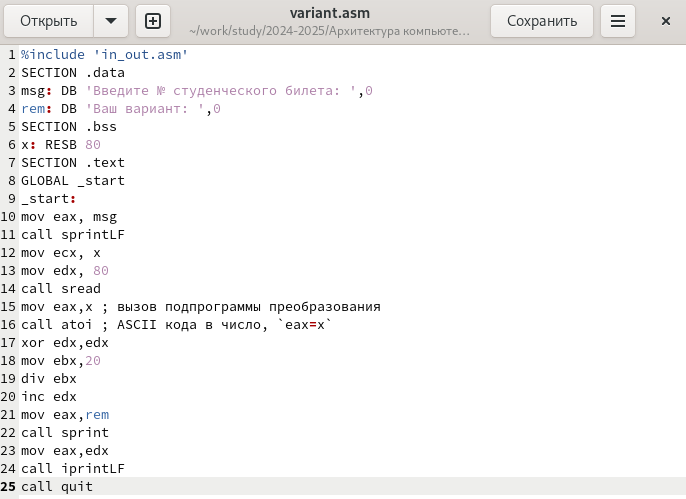


Рис. 18: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 19).

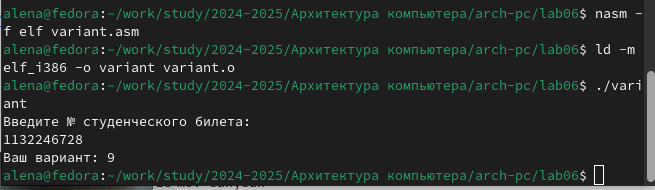


Рис. 19: Запуск исполняемого файла

### 4.2.1 Ответы на вопросы

1. За вывод на экран сообщения ‘Ваш вариант:’ отвечают строки:

mov eax,rem  
call sprint

1. Инструкция mov ecx,x используется, чтобы положить адрес вводимой строки 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,которая 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 отвчают строки:

mov eax,edx  
call iprintLF

## 4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm (рис. 20).

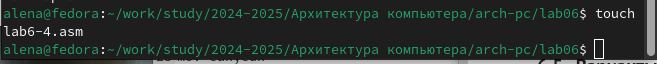


Рис. 20: Создание файла

Ввожу в созданный файл текст программы для вычисления 10+(31\*x-5) (вариант №9) (рис. 21).

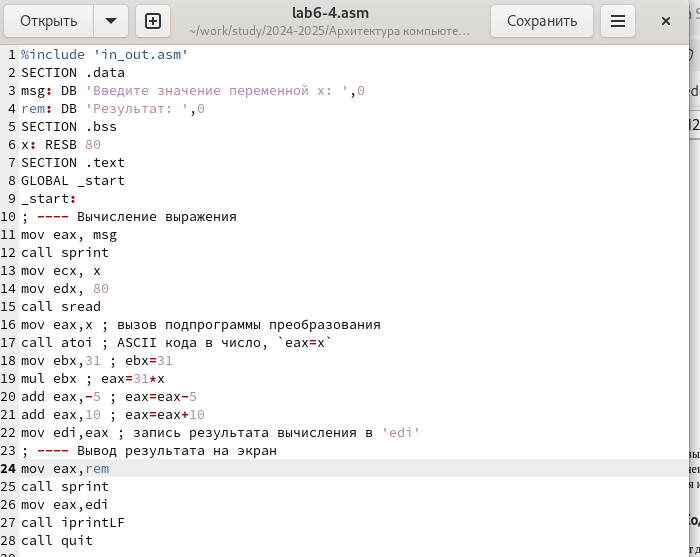


Рис. 21: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 22).

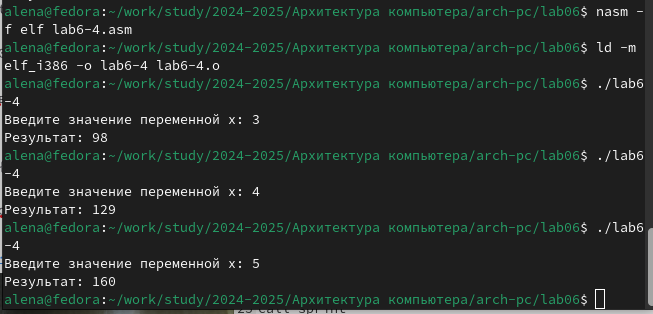


Рис. 22: Запуск исполняемого файла

Программа сработала верно

\*\*Листинг программы для вычисления выражения 10+(31\*x-5)

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите значение переменной x: ',0  
rem: DB 'Результат: ',0  
SECTION .bss  
x: RESB 80  
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`  
mov ebx,31 ; ebx=31  
mul ebx ; eax=31\*x  
add eax,-5 ; eax=eax-5  
add eax,10 ; eax=eax+10  
mov edi,eax ; запись результата вычисления в 'edi'  
; ---- Вывод результата на экран  
mov eax,rem  
call sprint  
mov eax,edi  
call iprintLF  
call quit

# 5 Выводы

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

# Список литературы

1. [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..pdf)