Отчет по лабораторной работе №5

Архитектура компьютера

Николенко Анна Николаевна

Содержание

# 1 Цель работы

Цель данной лабораторной работы - освоение арифметческих инструкций 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 операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти.

Основные способы 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.

Арифметические операции в NASM: \* Целочисленное сложение - add. \* Целочисленное вычитание - sub. \* Команды инкремент(inc)- прибавление единицы к операнду и декремент(dec)- вычитание единицы. Они выгодны тем, что они занимают меньше места, чем соответствующие команды сложения и вычитания. \* Команда изменения знака операнда - neg. \* Команды умножения - mul (для беззнакового умножения) и imul (для знакового умножения). \* Команды деления div и idiv.

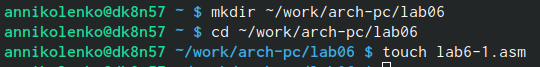
Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном 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 результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно

* iprint – вывод на экран чисел в формате ASCII.
* iprintLF – работает аналогично iprint, но при выводе на экран после числа добавляет к символ перевода строки.

# 4 Выполнение лабораторной работы

## 4.1 Символьные и численные данные в NASM

Создаю каталог для программам лабораторной работы No 6, перехожу в него и создаю файл lab6-1.asm (рис. [??]).



Создание каталога, переход в этот каталог и создание файла

Копирую в текущий каталог файл in\_out.asm с помощью команды cp, т.к. он будет использоваться в других программах (рис. [??]).

Копироавние файла

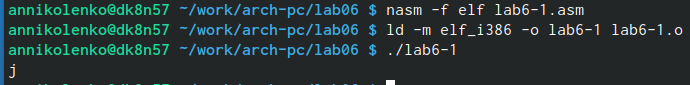
Копироавние файла

Открываю файл lab6-1.asm и ввожу в него программу вывода значения регистра eax (рис. [??]).



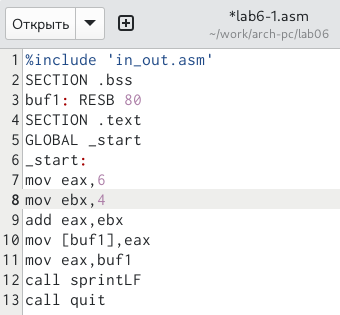
Ввод программы файл (вывода значения регистра eax)

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



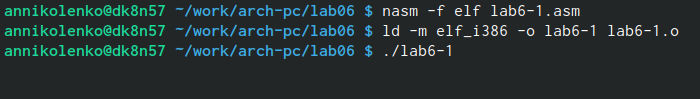
Создание исполняемого файла и его запуск

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. [??]).



Редактирование файла

Создаю новый исполняемый файл программы и запускаю его. Выводится символ с кодом 10, это символ перевода строки. Этот символ не отображается при выводе на экран (рис. [??]).



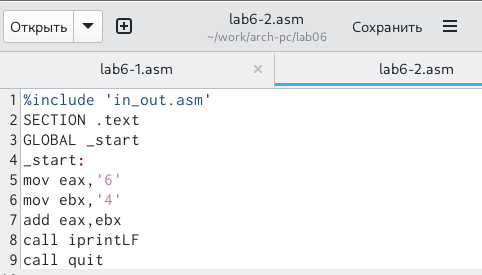
Редактирование файла

Создаю новый файл lab6-2.asm с помощью команды touch (рис. [??]).

Создание файла

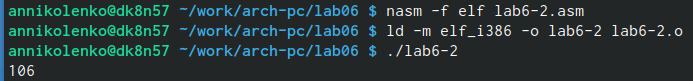
Создание файла

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



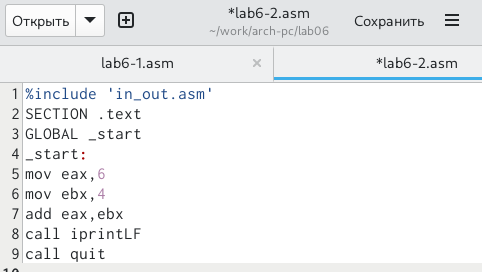
Редактирование файла

Создаю и запускаю исполняемый файл lab6-2. Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4” (рис. [??]).



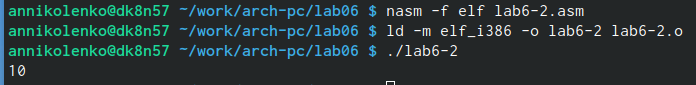
Запуск исполняемого файла

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



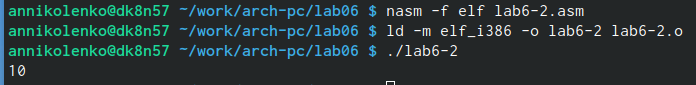
Редактирование файла

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



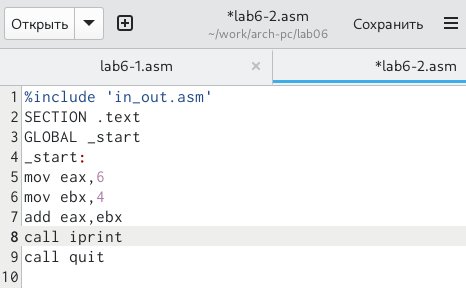
Запуск исполняемого файла

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



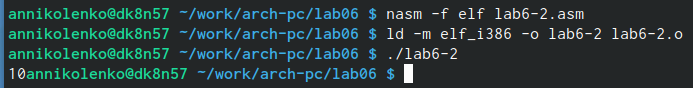
Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. [??]).



Редактирование файла

Создаю и запускаю новый исполняемый файл. 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. [??]).



Запуск исполняемого файла

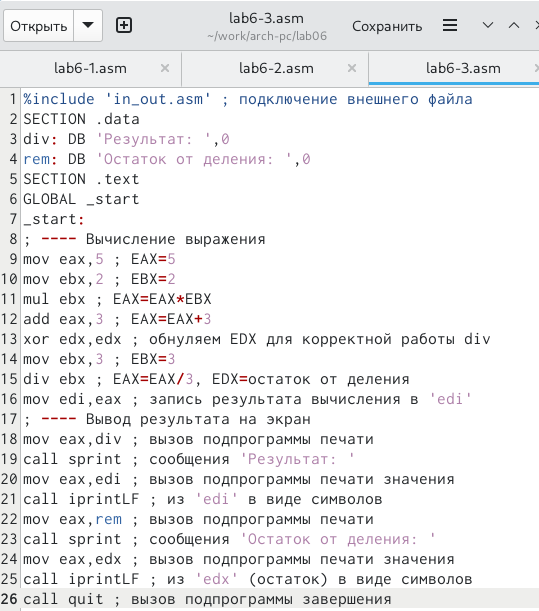
## 4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. [??]).

Создание файла

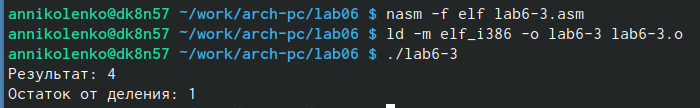
Создание файла

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



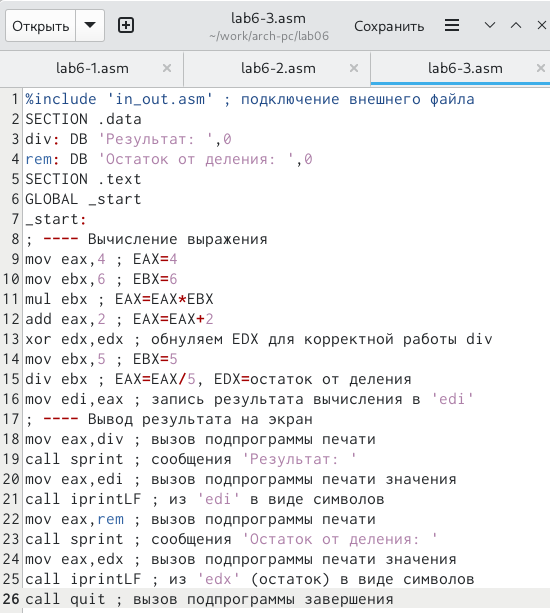
Редактирование файла

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



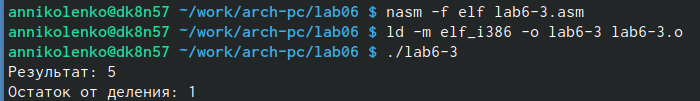
Запуск исполняемого файла

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



Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 19). Провериала его работту (корректность ответа), решив уравнение самостоятельно (рис. [??]).



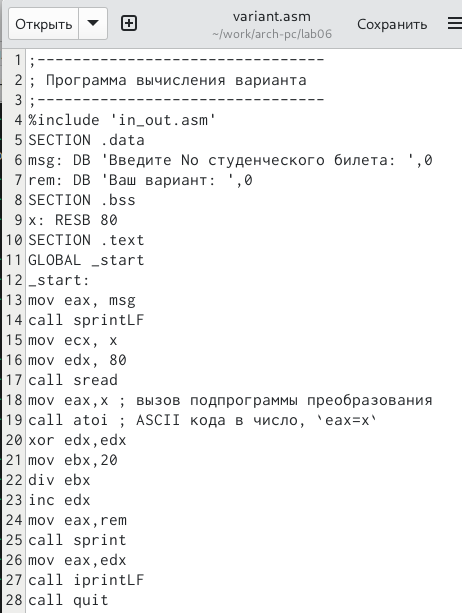
Запуск исполняемого файла

Создаю файл variant.asm с помощью команды touch (рис. [??]).

Создание файла

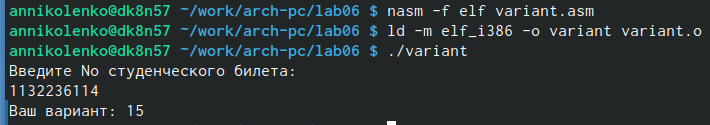
Создание файла

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



Редактирование файла

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 15 (рис. [??]).



Запуск исполняемого файла

# 5 Ответы на вопросы

1. За вывод сообщения “Ваш вариант” отвечают строки кода: mov eax,rem call sprint
2. 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
3. call atoi - вызов подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
4. 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
5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
6. Инструкция inc edx прибавляет 1 к значению регистра edx
7. За вывод на экран результатов вычислений отвечают строки: mov eax,edx call iprintLF

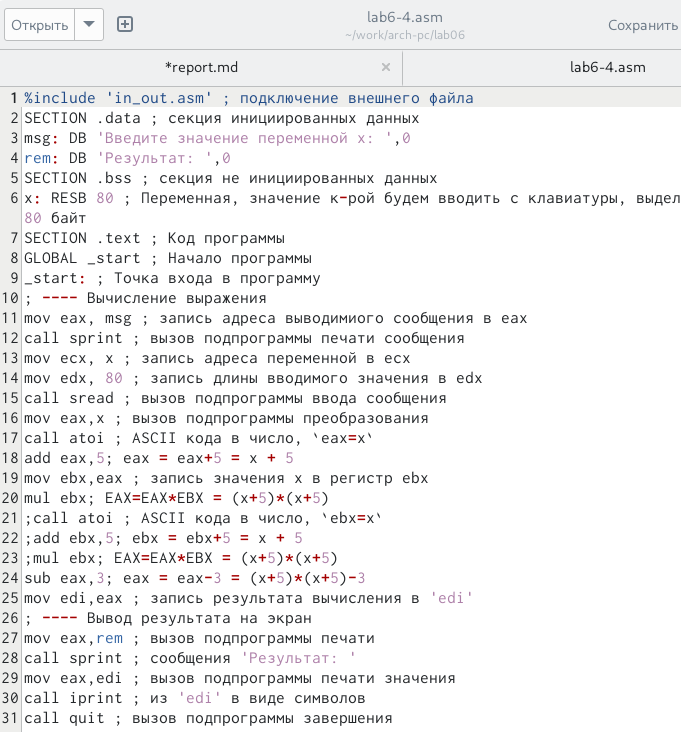
# 6 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью утилиты touch (рис. [??]).

Создание файла

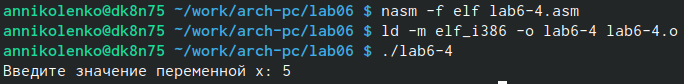
Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения (5 + x)^2 - 3 (^ - возведение в степень) (Вариант 15) (рис. [??]).



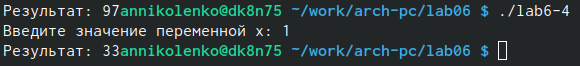
Написание программы

Создаю и запускаю исполняемый файл. При вводе значения 5, вывод - 97. (рис. [??]).



Запуск исполняемого файла

Повторный запуск исполняемого файла с другим занчением x=1, вывод - 33 (рис. [??]).



Запуск исполняемого файла

# 7 Листинги программ

1. Программа вывода значения регистра eax

%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 sprintLF call quit

1. Программа вывода значения регистра eax %include ‘in\_out.asm’ SECTION .text GLOBAL \_start \_start: mov eax,‘6’ mov ebx,‘4’ add eax,ebx call iprintLF call quit
2. Программа вычисления выражения 𝑓(𝑥) = (5 ∗ 2 + 3)/3 ;——————————– ; Программа вычисления выражения ;——————————– %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,5 ; EAX=5 mov ebx,2 ; EBX=2 mul ebx ; EAX=EAX\*EBX add eax,3 ; EAX=EAX+3 xor edx,edx ; обнуляем EDX для корректной работы div mov ebx,3 ; 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 завершения
3. Программа вычисления вычисления варианта задания по номеру студенческого билета ;——————————– ; Программа вычисления варианта ;——————————– %include ‘in\_out.asm’ SECTION .data msg: DB ‘Введите No студенческого билета:’,0 rem: DB ‘Ваш вариант:’,0 SECTION .bss x: RESB 80 SECTION .text GLOBAL \_start \_start: mov eax, msg call sprintLF mov ecx, x mov edx, 80 call sread mov eax,x ; вызов подпрограммы преобразования call atoi ; ASCII кода в число, eax=x xor edx,edx mov ebx,20 div ebx inc edx mov eax,rem call sprint mov eax,edx call iprintLF call quit
4. Программа для вычисления значения выражения (5 + x) ^ 2 − 3. %include ‘in\_out.asm’ ; подключение внешнего файла SECTION .data ; секция инициированных данных msg: DB ‘Введите значение переменной х:’,0 rem: DB ‘Результат:’,0 SECTION .bss ; секция не инициированных данных x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 байт SECTION .text ; Код программы GLOBAL \_start ; Начало программы \_start: ; Точка входа в программу ; —- Вычисление выражения mov eax, msg ; запись адреса выводимиого сообщения в eax call sprint ; вызов подпрограммы печати сообщения mov ecx, x ; запись адреса переменной в ecx mov edx, 80 ; запись длины вводимого значения в edx call sread ; вызов подпрограммы ввода сообщения mov eax,x ; вызов подпрограммы преобразования call atoi ; ASCII кода в число, eax=x add eax,5; eax = eax+5 = x + 5 mov ebx,eax ; запись значения x в регистр ebx mul ebx; EAX=EAX*EBX = (x+5)*(x+5) ;call atoi ; ASCII кода в число, ebx=x ;add ebx,5; ebx = ebx+5 = x + 5 ;mul ebx; EAX=EAX*EBX = (x+5)*(x+5) sub eax,3; eax = eax-3 = (x+5)\*(x+5)-3 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 iprint ; из ‘edi’ в виде символов call quit ; вызов подпрограммы завершения

# 8 Выводы

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

# Список литературы