Отчёт по лабораторной работе №6

дисциплина: Архитектура компьютера

Аносов Даниил Игоревич

Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

1. Написать программу вычисления выражения = . Программа должна выводить выражение для вычисления, выводить запрос на ввод значения , вычислять заданное выражение в зависимости от введенного, выводить результат вычислений. Вид функции выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений и из 6.3.

# 3 Выполнение лабораторной работы

Откроем терминал и создадим каталог для программ лабораторной работы №6 (рис. 1).

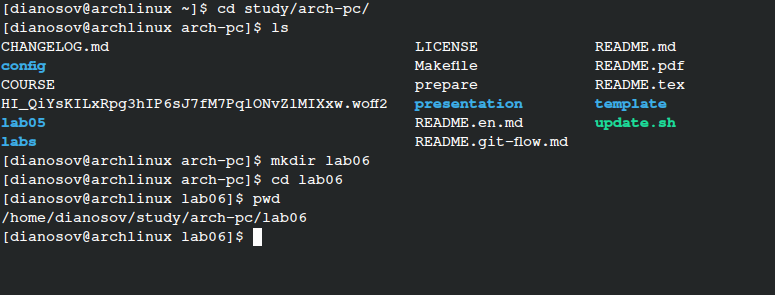


Рис. 1: Создание каталога для программ

Создадим файл *lab6-1.asm* и скопируем из каталога предыдущей лабораторной работы файл *in\_out.asm* с полезными подпрограммами (рис. 2).

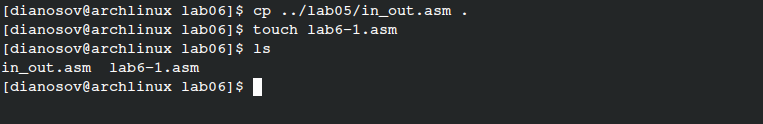


Рис. 2: Подготовка файлов

## 3.1 Вывод чисел в терминал в NASM

Скопируем код из предложенного листинга в файл *lab6-1.asm* (рис. 3).

В данной программе в регистр eax записывается символ 6 (mov eax, '6'), в регистр *ebx* символ 4 (mov ebx,'4'). Далее к значению в регистре *eax* прибавляем значение регистра *ebx* (add eax,ebx, результат сложения запишется в регистр eax). Далее выводим результат. Так как для работы функции *sprintLF* в регистр eax должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра *eax* в переменную *buf1* (mov [buf1], eax), а затем запишем адрес переменной *buf1* в регистр *eax* (mov eax, buf1) и вызовем функцию *sprintLF*.



Рис. 3: Vim с файлом lab6-1.asm

Скомпилируем и запустим программу *lab6-1.asm* (рис. 4). В данном случае при выводе значения регистра *eax* мы ожидаем увидеть число 10. Однако результатом будет символ *j*. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда add eax,ebx запишет в регистр *eax* сумму кодов – 01101010 (106), что в своюочередь является кодом символа *j*.

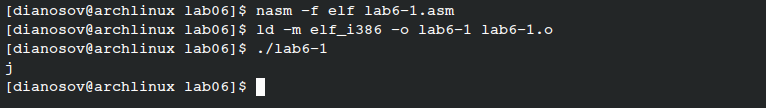


Рис. 4: Компиляция и первый запуск программы

Изменим текст программы и вместо символов запишем в регистры числа. Исправим текст программы следующим образом: заменим строки

mov eax,'6'  
mov ebx,'4'

на строки

mov eax,6  
mov ebx,4

Снова скомпилируем программу и запустим её (рис. 5).

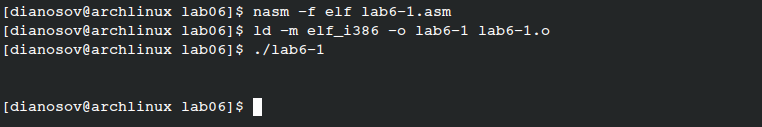


Рис. 5: Повторная компиляция и запуск программы

Как и в предыдущем случае при исполнении программы мы не получили число 10. В данном случае должен выводиться символ с кодом 10. Он не отображается в терминале.

В этом же каталоге создадим файл *lab6-1.asm*. В него вставим код из предложенного листинга, использующий команду *iprintLF* (рис. 6).

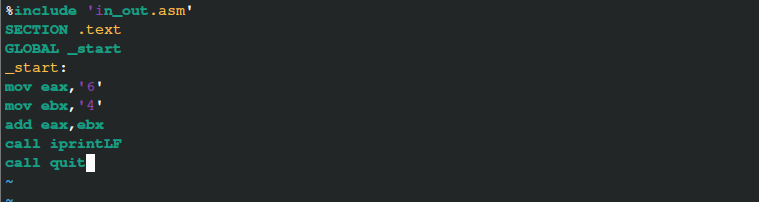


Рис. 6: Редактирование файла *lab6-2.asm*

Скомпилируем и запустим новую программу (рис. 7). В результате работы программы мы получим число 106. В данном случае, как и в первом, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако, в отличии от программы из листинга 6.1, функция *iprintLF* позволяет вывести число, а не символ, кодом которого является это число.

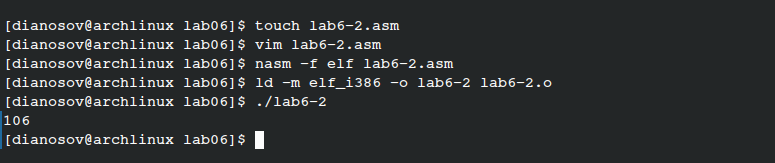


Рис. 7: Компиляция и запуск новой программы

Аналогично предыдущему примеру изменим символы на числа. Заменим строки

mov eax,'6'  
mov ebx,'4'

на строки

mov eax,6  
mov ebx,4

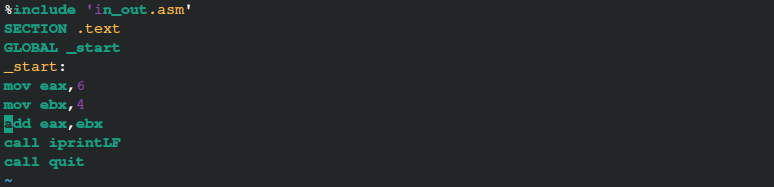


Рис. 8: Редактирование файла *lab6-2.asm*

Скомпилируем и запустим программу *lab6-2.asm* (рис. 9). Отметим, что команда *iprintLF* не просто выводит число, а ещё и переводит строку после него. Команда *iprint*, напротив, только выводит значение из *eax*.

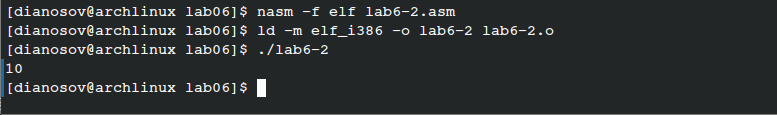


Рис. 9: Повторная компиляция и запуск новой программы

## 3.2 Арифметические операции в NASM

### 3.2.1 Вычисление значения выражения

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения .

Создадим в рабочем каталоге файл *lab6-3.asm* и запишем в него код из предложенного листинга (рис. 10).

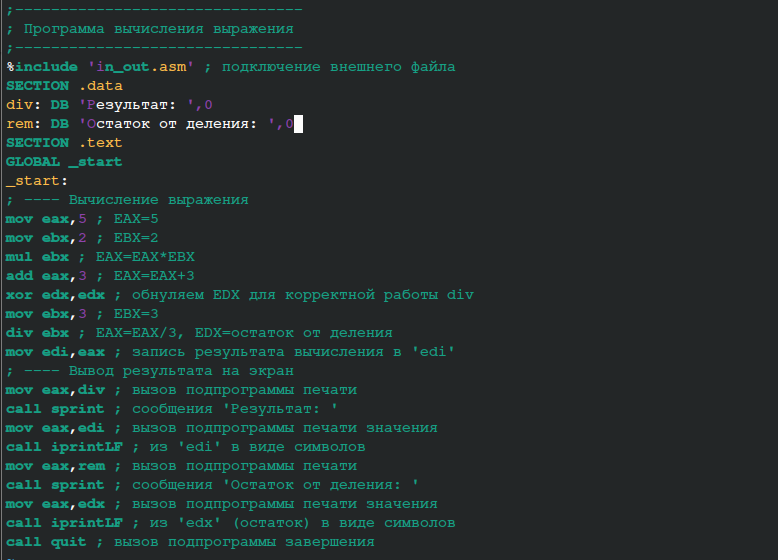


Рис. 10: Редактирование третьей программы

Скомпилируем и запустим программу *lab6-3.asm* (рис. 11).

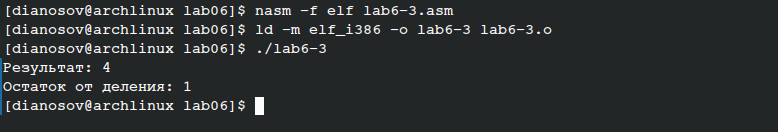


Рис. 11: Компиляция и запуск третьей программы

Теперь сделаем так, чтобы программа вычисляла значение выражения . Откроем файл с кодом в редакторе **Vim** и отредактируем код (рис. 12).

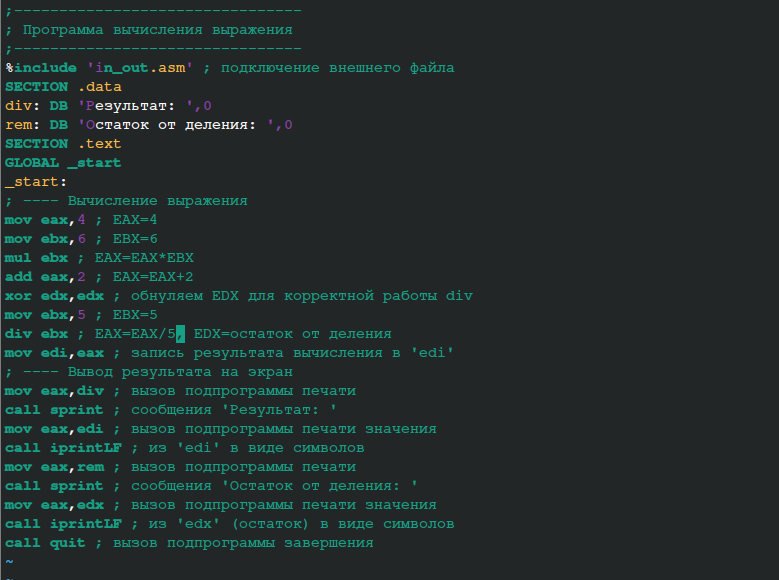


Рис. 12: Изменение кода третьей программы

Снова скомпилируем и запустим программу *lab6-3.asm* (рис. 13).

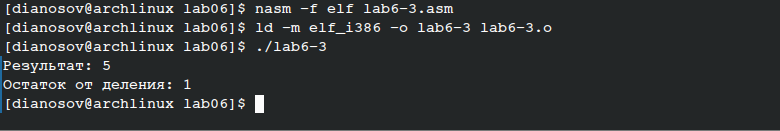


Рис. 13: Компиляция и запуск третьей программы после редактирования

Видим, что программа работает корректно. Действительно, .

### 3.2.2 Программа вычисления номера варианта по номеру студенческого билета

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму: - вывести запрос на введение № студенческого билета - вычислить номер варианта по формуле: , где – номер студенческого билета. - вывести на экран номер варианта.

Создадим в рабочем каталоге файл *variant.asm* (рис. 14).

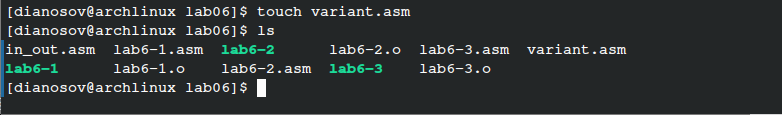


Рис. 14: Создание файла *variant.asm*

В новую программу введём код из предложенного листинга. (рис. 15).

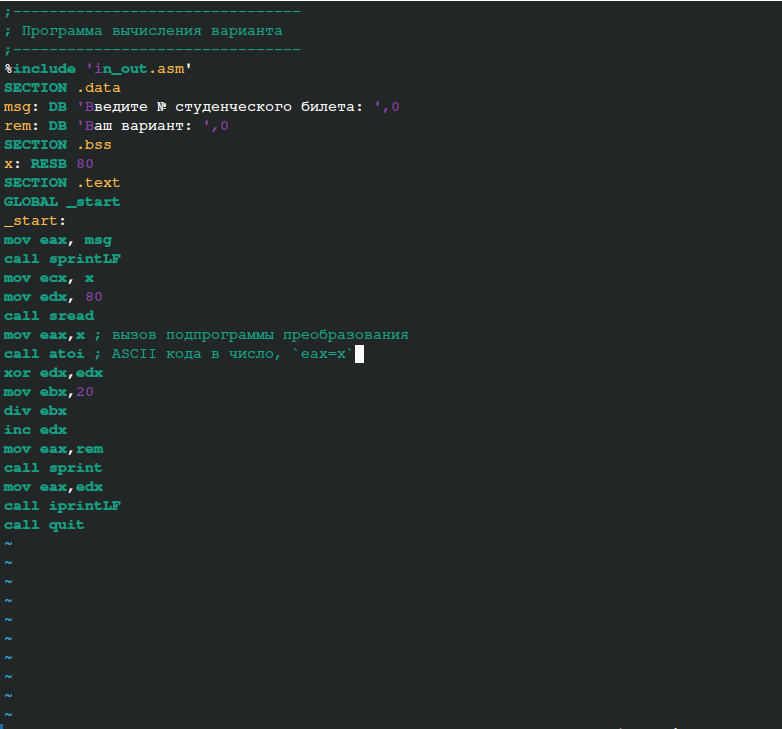


Рис. 15: Редактирование файла *variant.asm*

Проведем компиляцию нового файла и проверим его работу. (рис. 16).

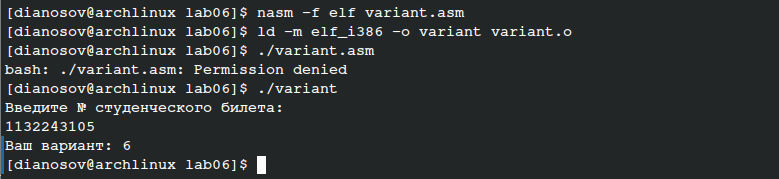


Рис. 16: Компиляция и запуск *variant.asm*

Программа верно вычисляет вариант. Действительно, . Ответы на вопросы: 1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? За это отвечают строки:

mov eax, msg  
call sprintLF

1. Для чего используется следующие инструкции?

mov ecx, x  
mov edx, 80  
call sread

Они отвечают за считывание ввода из терминала. Первая строчка говорит, что считанное значение будет помещено в x. Вторая строчка задает максимальный объём данных для считывания. Третья - вызывает функцию *sread* считывания строки из терминала.

1. Для чего используется инструкция “call atoi”?

Для преобразования значения в регистре eax в число.

1. Какие строки листинга 6.4 отвечают за вычисления варианта?

mov ebx,20  
div ebx  
inc edx

1. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

Остаток от деления помещается в регистр edx.

1. Для чего используется инструкция “inc edx”?

Эта инструкция используется для увеличения значения в регистре edx на 1.

1. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

mov eax,rem  
call sprint  
mov eax,edx  
call iprintLF

# 4 Задание для самостоятельной работы

Полученный номер варианта - 6, как указано ранее. Выражение:

Скопируем файл lab6-3.asm и дадим новому файлу название task.asm. (рис. 17). В новом файле сделаем нужные изменения (код прокомментирован).

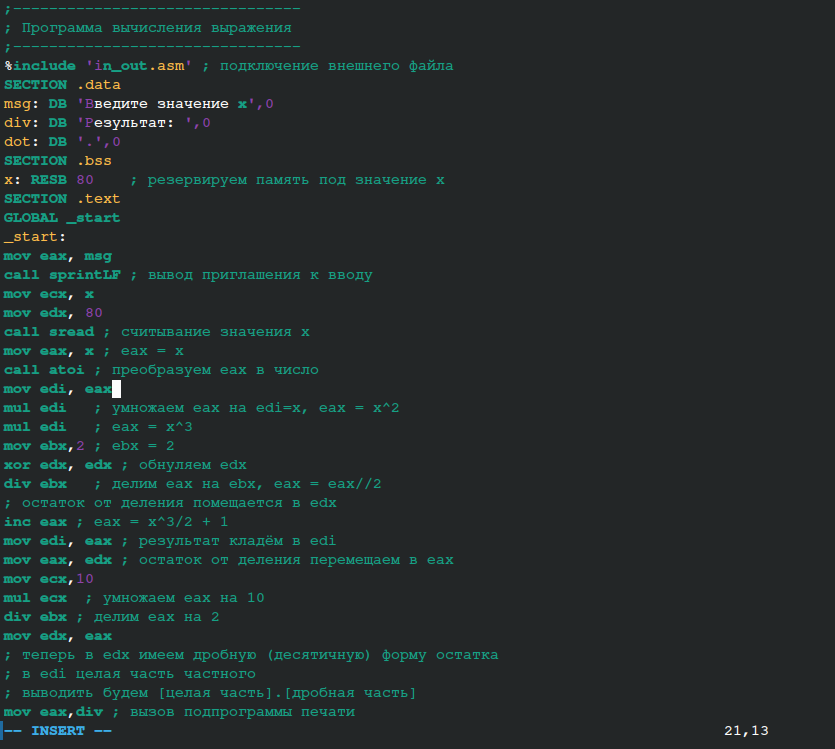
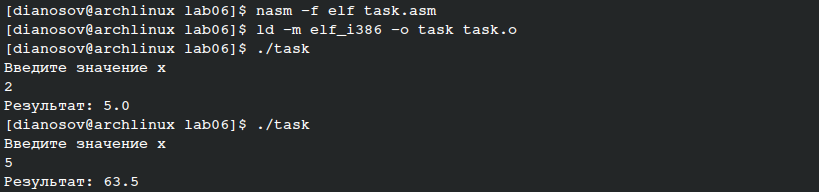


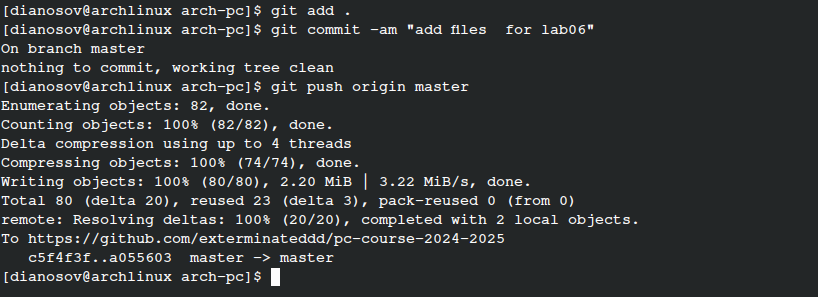
Рис. 17: Редактирование файла *task.asm*

Проверим коррекность работы программмы для (рис. **¿fig:018?**).

 Программа работает правильно.

Ниже приведен весь код новой программы с подробными комментариями.

%include 'in\_out.asm' ; подключение внешнего файла  
SECTION .data  
msg: DB 'Введите значение x',0  
div: DB 'Результат: ',0  
dot: DB '.',0  
SECTION .bss  
x: RESB 80 ; резервируем память под значение x  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax, msg  
call sprintLF ; вывод приглашения к вводу  
mov ecx, x  
mov edx, 80  
call sread ; считывание значения x  
mov eax, x ; eax = x  
call atoi ; преобразуем eax в число  
mov edi, eax  
mul edi ; умножаем eax на edi=x, eax = x^2  
mul edi ; eax = x^3  
mov ebx,2 ; ebx = 2  
xor edx, edx ; обнуляем edx  
div ebx ; делим eax на ebx, eax = eax//2  
; остаток от деления помещается в edx  
inc eax ; eax = x^3/2 + 1  
mov edi, eax ; результат кладём в edi  
mov eax, edx ; остаток от деления перемещаем в eax  
mov ecx,10  
mul ecx ; умножаем eax на 10  
div ebx ; делим eax на 2  
mov edx, eax  
; теперь в edx имеем дробную (десятичную) форму остатка  
; в edi целая часть частного  
; выводить будем [целая часть].[дробная часть]  
mov eax,div ; вызов подпрограммы печати  
call sprint ; сообщения 'Результат: '  
mov eax,edi ; вызов подпрограммы печати значения  
call iprint ; из 'edi' в виде символов  
mov eax,dot ; точка  
call sprint ; печать точки  
mov eax,edx ; вызов подпрограммы печати значения  
call iprintLF ; из 'edx' (остаток) в виде символов  
call quit ; вызов подпрограммы завершения

Задание выполнено, загрузим новую версию проекта курса на GitHub. 

# 5 Выводы

В ходе выполнения лабораторной работы были освоены арифметические инструкции языка ассемблера NASM.