Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Алексеев Илья Сергеевич

Содержание

Список иллюстраций

# 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. - 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 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 утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. 1). Перехожу в созданный каталог с помощью утилиты cd.

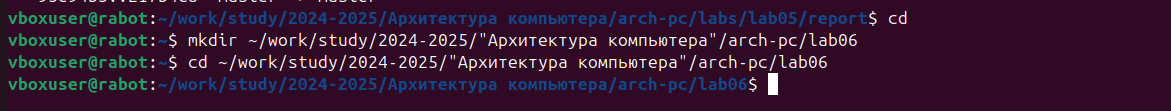


Рис. 1: Создание директории

С помощью утилиты touch создаю файл lab6-1.asm (рис. 2).

Рис. 2: Создание файла

Рис. 2: Создание файла

Копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. 3).

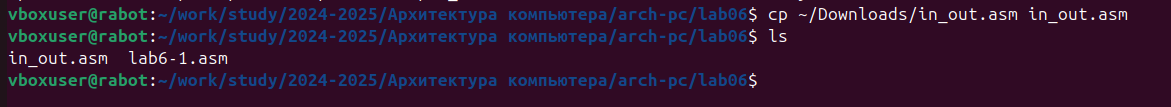


Рис. 3: Создание копии файла

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. 4).

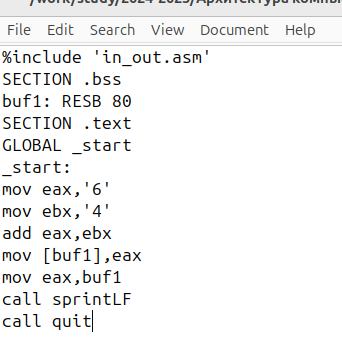


Рис. 4: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 5). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

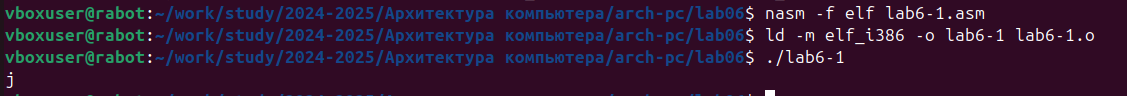


Рис. 5: Запуск исполняемого файла

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

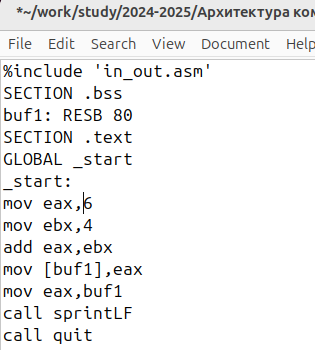


Рис. 6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 7). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

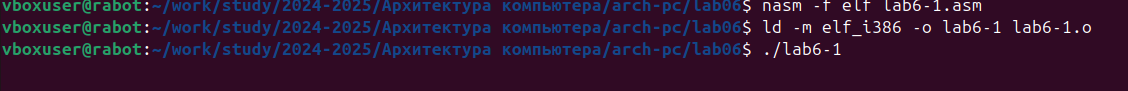


Рис. 7: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 8).

Рис. 8: Создание файла

Рис. 8: Создание файла

Ввожу в файл текст другойпрограммы для вывода значения регистра eax (рис. 9).

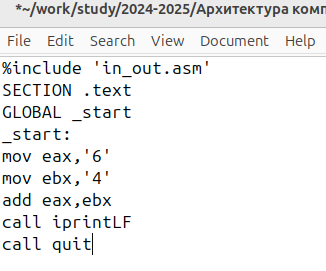


Рис. 9: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 10). 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”.

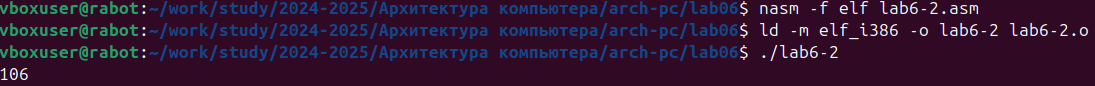


Рис. 10: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 11).

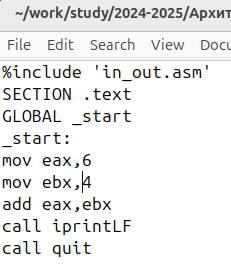


Рис. 11: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 12).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

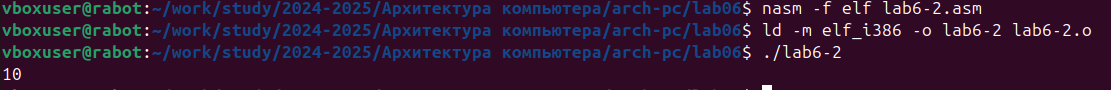


Рис. 12: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 13).

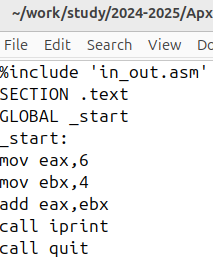


Рис. 13: Редактирование файла

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

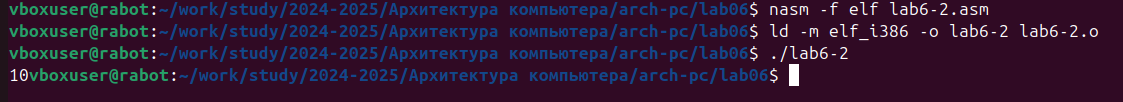


Рис. 14: Запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. 15).

Рис. 15: Создание файла

Рис. 15: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения f(x) = (5 \* 2 + 3)/3 (рис. 16).

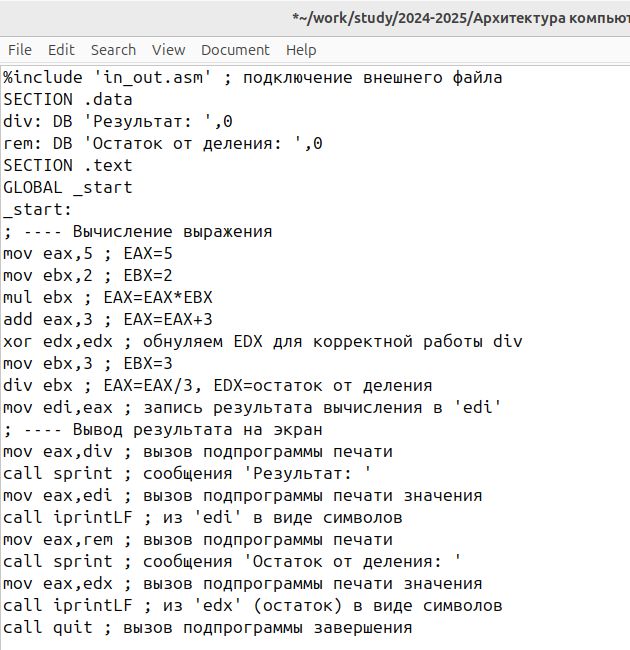


Рис. 16: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 17).

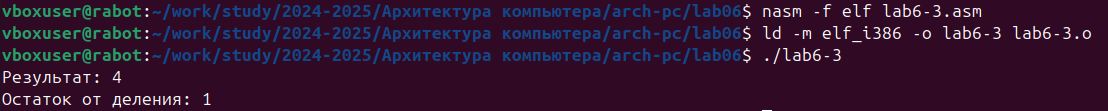


Рис. 17: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5 (рис. 18).

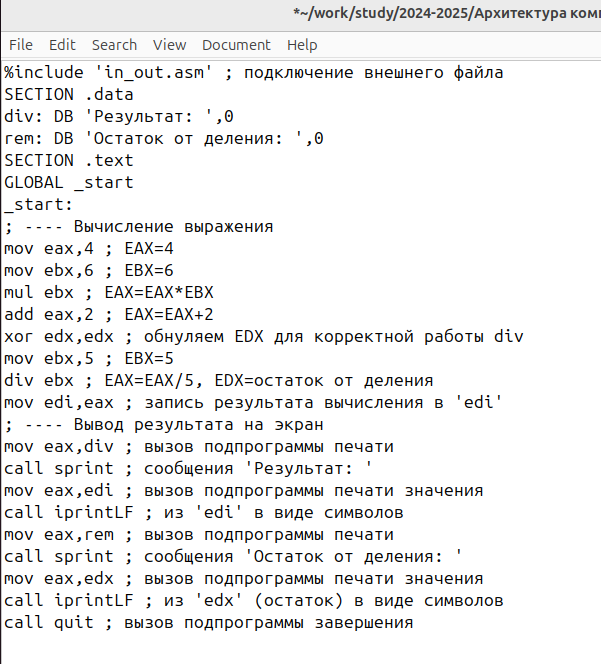


Рис. 18: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 19). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

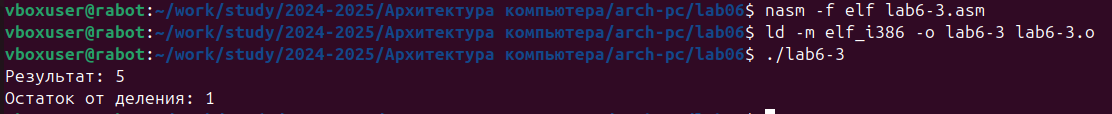


Рис. 19: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 20).

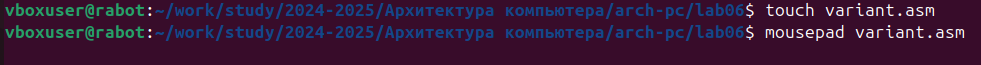


Рис. 20: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 21).

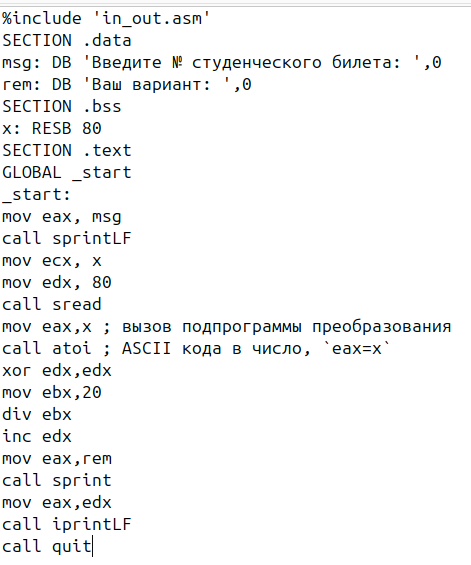


Рис. 21: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 22). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 8.

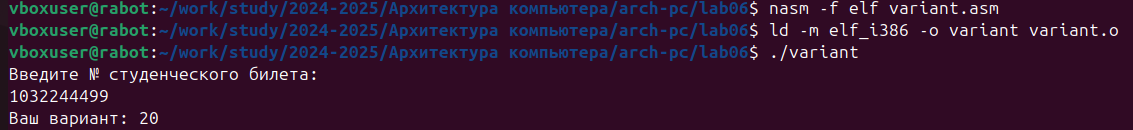


Рис. 22: Запуск исполняемого файла

### 4.2.1 Ответы на вопросы по программе

1. Строка “mov eax,rem” и строка “call sprint” отвечают за вывод на экран сообщения ‘Ваш вариант:’.
2. Эти инструкции используются для чтения строки с вводом данных от поль- зователя. Начальный адрес строки сохраняется в регистре ecx, а количество символов в строке (максимальное количество символов, которое может быть считано) сохраняется в регистре edx. Затем вызывается процедура sread, которая выполняет чтение строки.
3. Инструкция “call atoi” используется для преобразования строки в целое число. Она принимает адрес строки в регистре eax и возвращает полученное число в регистре eax.
4. Строка “xor edx,edx” обнуляет регистр edx перед выполнением деления. Строка “mov ebx,20” загружает значение 20 в регистр ebx. Строка “div ebx” выполняет деление регистра eax на значение регистра ebx с сохранением частного в регистре eax и остатка в регистре edx.
5. Остаток от деления записывается в регистр edx.
6. Инструкция “inc edx” используется для увеличения значения в регистре edx на 1. В данном случае, она увеличивает остаток от деления на 1.
7. Строка “mov eax,edx” передает значение остатка от деления в регистр eax. Строка “call iprintLF” вызывает процедуру iprintLF для вывода значения на экран вместе с переводом строки.

## 4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью утилиты touch (рис. 23).

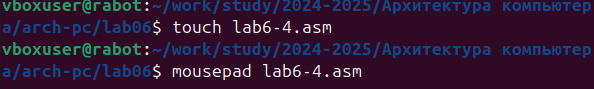


Рис. 23: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения x^3\*1/3+21 (рис. 24). Это выражение было под вариантом 8.

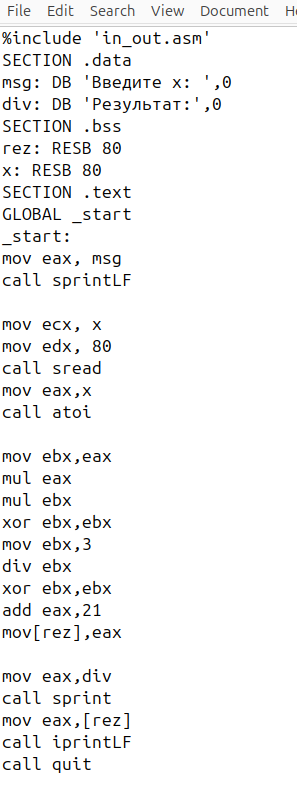


Рис. 24: Написание программы

Создаю и запускаю исполняемый файл (рис. 25). При вводе значения 1, вывод - 21.

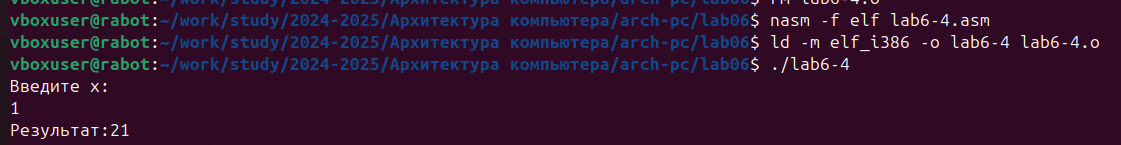


Рис. 25: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. 26). Программа отработала верно.

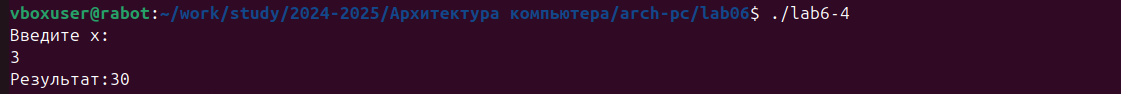


Рис. 26: Запуск исполняемого файла

\*\*Листинг 4.1. Программа для вычисления значения выражения x^3\*1/3+21.\*\*

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите x: ',0  
div: DB 'Результат:',0  
SECTION .bss  
rez: RESB 80  
x: RESB 80  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax, msg  
call sprintLF  
  
mov ecx, x  
mov edx, 80  
call sread  
mov eax,x  
call atoi  
  
mov ebx,eax  
mul eax  
mul ebx  
xor ebx,ebx  
mov ebx,3  
div ebx  
xor ebx,ebx  
add eax,21  
mov[rez],eax  
  
mov eax,div  
call sprint  
mov eax,[rez]  
call iprintLF  
call quit

# 5 Выводы

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

# 6 Список литературы

1. [Таблица ASCII](https://www.rapidtables.com/code/text/ascii-table.html)