Шаблон отчёта по лабораторной работе

Простейший вариант

Сокирка Анна Константиновна

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Цель данной лабораторной работы - освоение арифметических 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

# 5 4.1 Символьные и численные данные в NASM

Создам каталог для программам лабораторной работы № 6, перейду в него и создам файл lab6-1.asm (рис. 1).

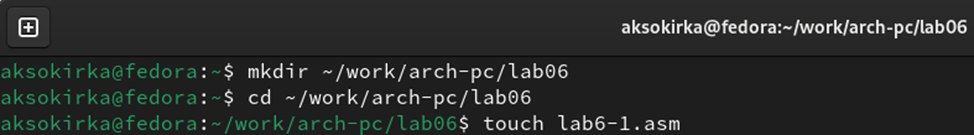


Рис. 1: Создание каталога

Введу в файл lab6-1.asm текст программы из листинга 6.1 (рис. 2).



Рис. 2: Ввод текста из листинга

Копирую файл in\_out.asm в файл lab06 (рис. 3).

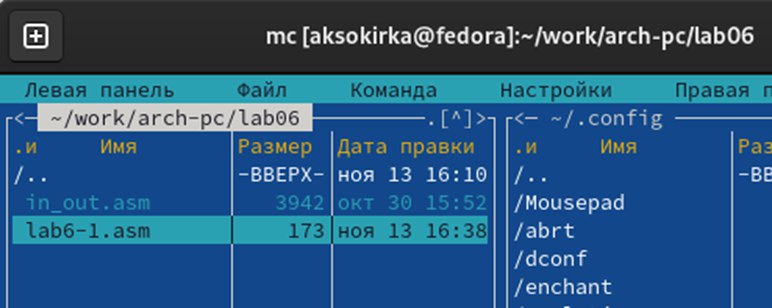


Рис. 3: Копирование файла

Создам исполняемый файл и запущу его (рис. 4).

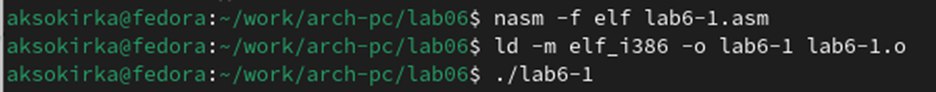


Рис. 4: Запуск исполняемого файла

Далее изменю текст программы и вместо символов, запишу в регистры числа. Исправлю текст программы (Листинг 6.1) (рис. 5).

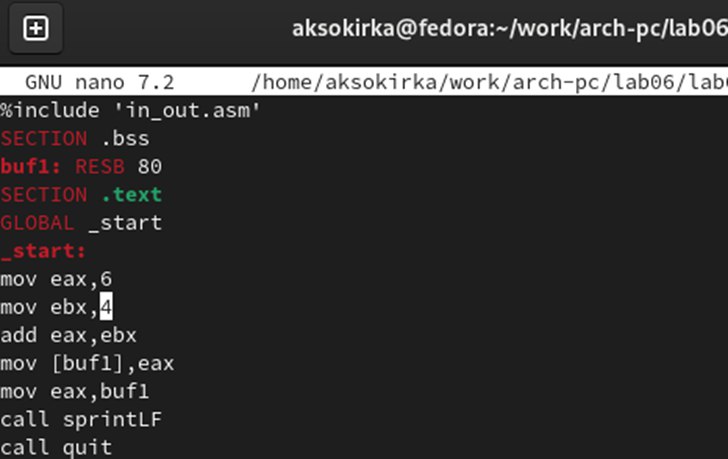


Рис. 5: Изменение текста программы

Создам исполняемый файл и запущу его.Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран (рис. 6).

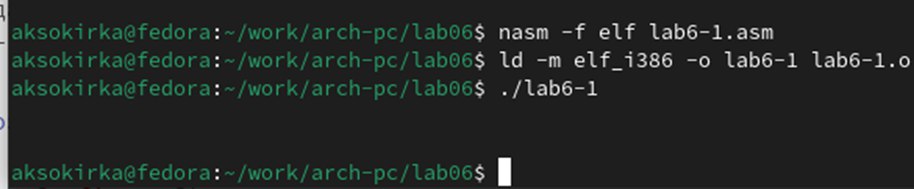


Рис. 6: Создание файла

Создам файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введу в него текст программы из листинга 6.2. (рис. 7).



Рис. 7: Создание файла

Создам исполняемый файл и запущу его.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. 8).

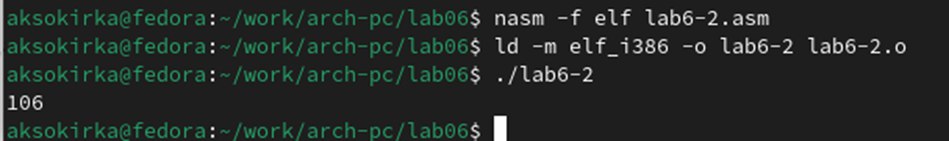


Рис. 8: Создание исполняемого файла

Аналогично предыдущему примеру изменю символы на числа (рис. 9).

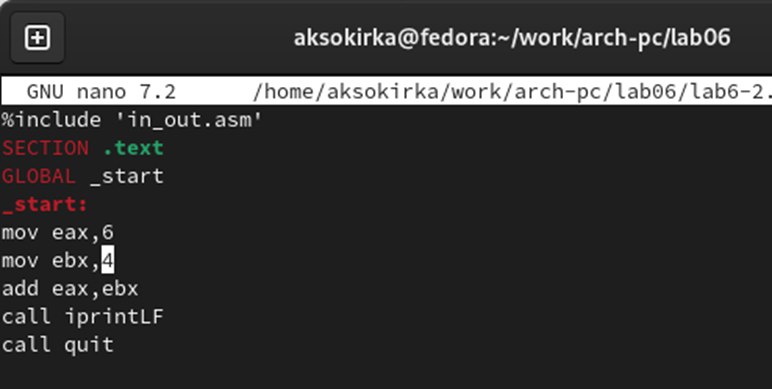


Рис. 9: Изменение текста программы

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

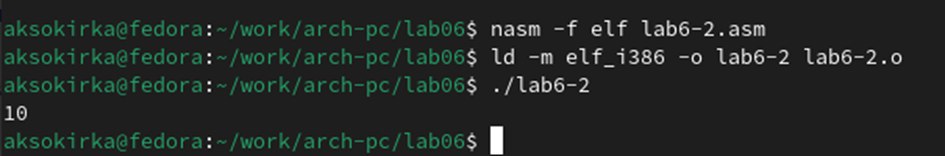


Рис. 10: Создание файла

Заменю функцию iprintLF на iprint. Создам исполняемый файл и запущу его. Вывод изменился. Функция iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF. (рис. 11).

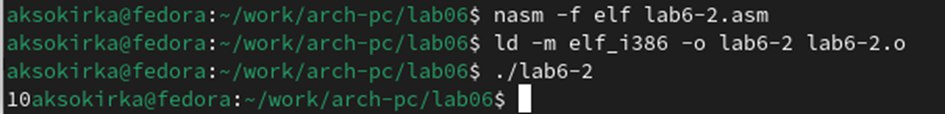


Рис. 11: Изменение текста программы

# 6 4.2 Выполнение арифметических операций в NASM

Создам файл lab6-3.asm в каталоге ~/work/arch-pc/lab06. Внимательно изучаю текст программы из листинга 6.3 и ввожу в lab6-3.asm (рис. 12).

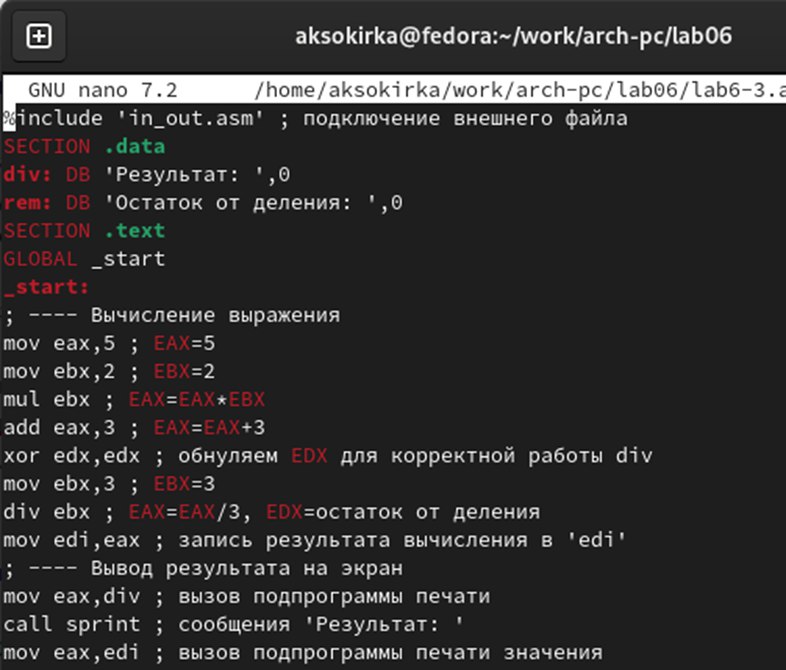


Рис. 12: Создание файла

Создам исполняемый файл и запущу его (рис. 13).

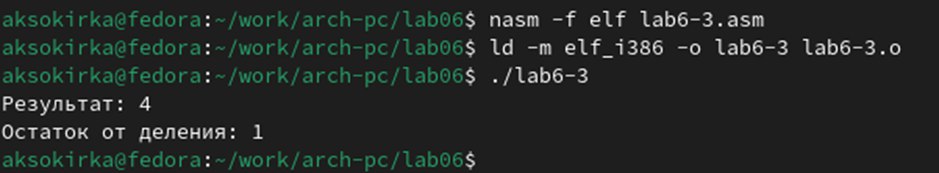


Рис. 13: Создание и запуск файла

Изменю текст программы для вычисления выражения 𝑓(𝑥) = (4 \* 6 + 2)/5. Создам исполняемый файл и проверю его работу (рис. 14).

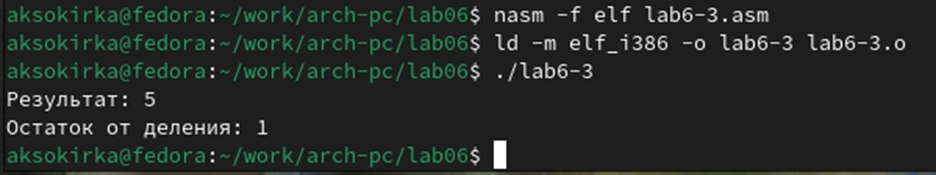


Рис. 14: Проверка исполняемого файла

Внимательно изучаю текст программы из листинга 6.4 и ввожу в файл variant.asm. Создам исполняемый файл и запущу его (рис. 15).



Рис. 15: Запуск программы

# 7 4.2.1 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода: mov eax,rem call sprint
2. 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
3. call atoi используется для вызова 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 увеличивает значение регистра edx на 1
7. За вывод на экран результатов вычислений отвечают строки: mov eax,edx call iprintLF

# 8 4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab7-4.asm (рис. 16).

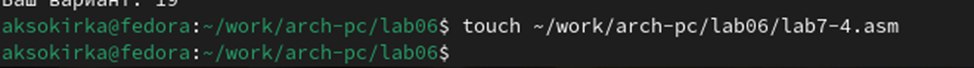


Рис. 16: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения (1/3*𝑥 + 5)*7. Это выражение было под вариантом 19 (рис. 17).

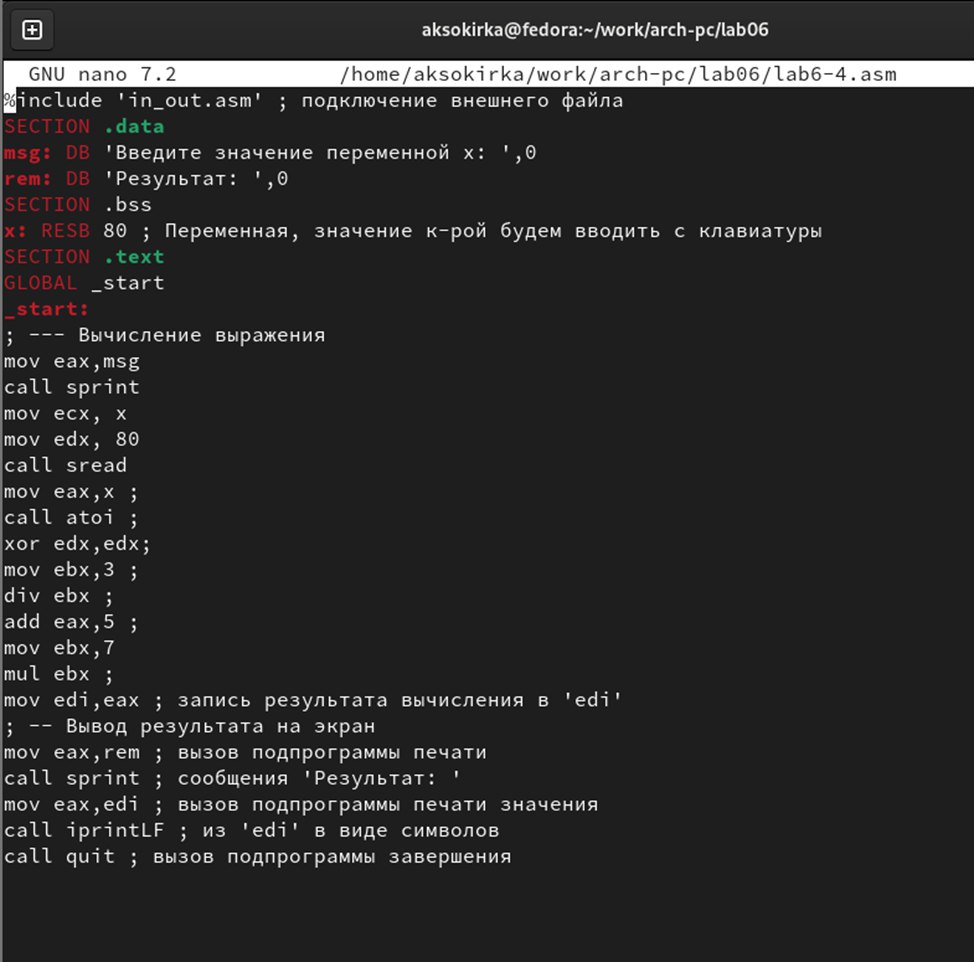


Рис. 17: Ввод текста программы

Создаю и запускаю исполняемый файл (рис. 18).

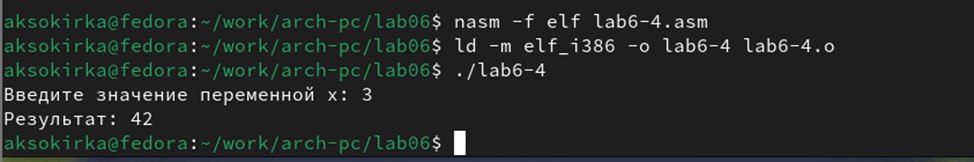


Рис. 18: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. 19).

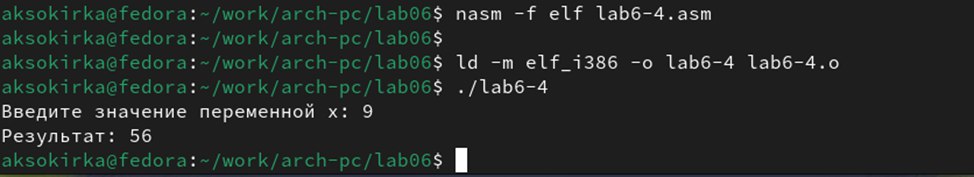


Рис. 19: Проверка работы исполняемого файла

# 9 Выводы

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

# 10 Список литературы

https://esystem.rudn.ru/pluginfile.php/2089086/mod\_resource/content/0/Лабораторная%20работа%20№6.%20Арифметические%20операции%20в%20NASM..pdf