Отчёт по лабораторной работе №6

НКАбд-03-25

Кулаженкова Яна Сергеевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель работа - освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Изучить инфинформацию об арифметических операциях в NASM
2. Выполнить лабораторную работу с использованием изученной информации
3. Выполнить задания для самостоятельной работы.

3 Теоретическое введение

В языке ассемблера NASM существуют три основных способа адресации операндов. 1. Регистровая адресация - операнды находятся в регистрах процессора (пример: mov ax, bx) 2. Непосредственная адресация - значение операнда задается непосредственно в команде (пример: mov ax, 2) 3. Адресация памяти - операнд указывает на адрес в памяти (пример: mov eax, [intg])

Команда mov eax, [intg] загружает значение из памяти, а mov eax, intg загружает адрес переменной.

В [**tbl**] приведены основные арифметические команды в NASM.

Таблица 3.1: Основные арифметические команды NASM {#tbl}

Арифметическая операция	Команда
Сложение	add <операнд1>, <операнд2>
Вычитание	sub <операнд1>, <операнд2>
Инкремент/декремент	inc <операнд>, dec <операнд>
Изменение знака	neg <операнд>
Умножение	mul (беззнаковое), imul (знаковое)
Деление	div (беззнаковое), idiv (знаковое)

Операции умножения и деления имеют особенности: один операнд задается явно, второй находится в регистрах AL/AX/EAX, а результат помещается в определенные пары регистров в зависимости от размера операндов.

Ввод и вывод данных осуществляются в символьном виде с использованием кодировки ASCII. Для корректной работы с числами необходимо преобразование между символьным и числовым представлением с помощью специализированных функций.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

Создадим каталог для программ лабораторной работы № 6. Внутри него создадим файл lab6-1.asm (рис. 4.1).

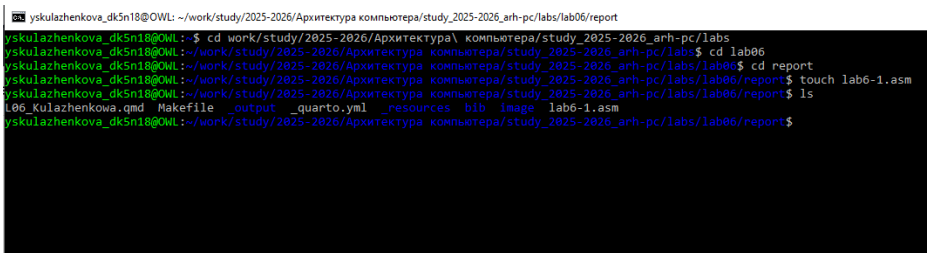

Рисунок 4.1: Начало работы с файлом

Рассмотрим примеры программ вывода символьных и численных значений. Для этого введем в наш файл следующий текст программы (рис. 4.2).

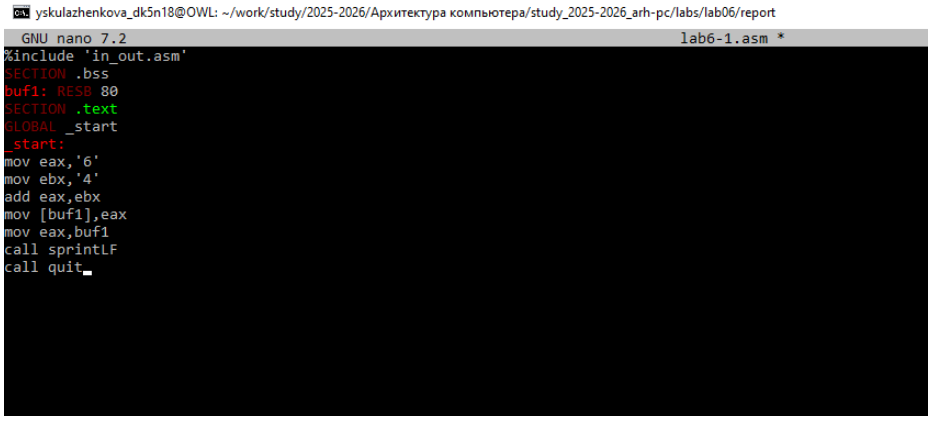

Рисунок 4.2: Подготовка файла

После создадим исполняемый файл и запустим его (рис. 4.3).

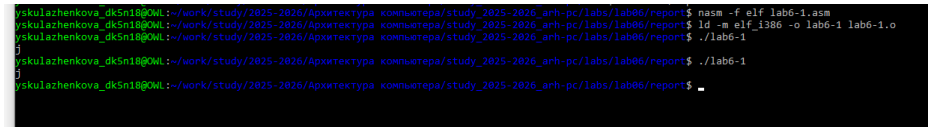

Рисунок 4.3: Исполняемый файл

Далее изменим текст программы и вместо символов, запишем в регистры числа (рис. 4.4).

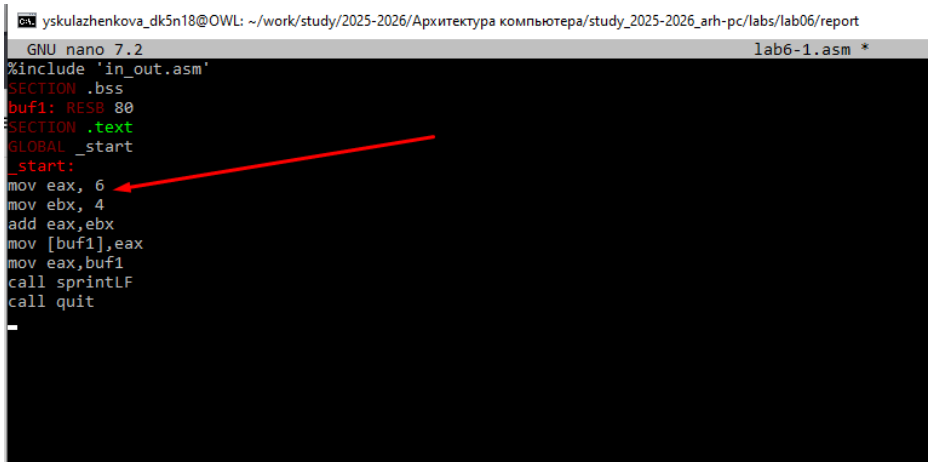

Рисунок 4.4: Изменим код

Снова создадим исполняемый файл. Проверим работу файла (рис. 4.5).

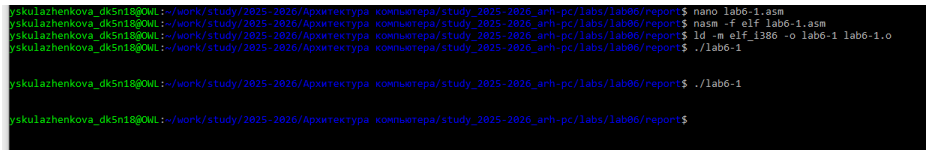

Рисунок 4.5: Запуск файла

Теперь преобразуем текст программы так, чтобы были использованы другие функции. Для этого создадим файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введем в него новый текст программы (рис. 4.6).

Рисунок 4.6: Второй файл

Продемонстрируем работу файла. Создадим исполняемый файл и запустим его (рис. 4.7).

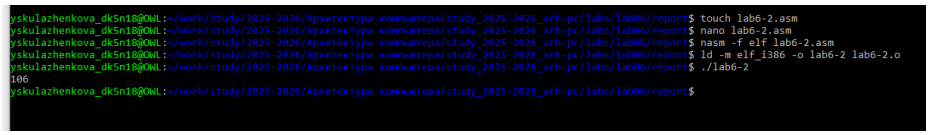

Рисунок 4.7: Запуск второго файла

Аналогично предыдущему примеру заменим символы на числа (рис. 4.8).

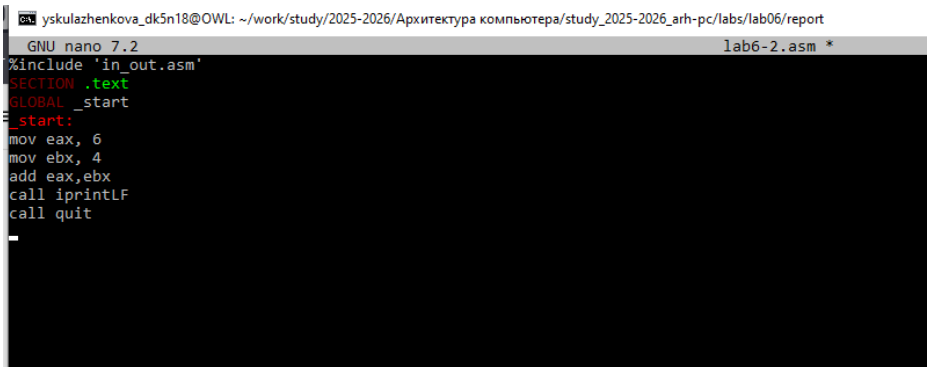

Рисунок 4.8: Новый код программы

Создадим исполняемый файл и запустим его (рис. 4.9).

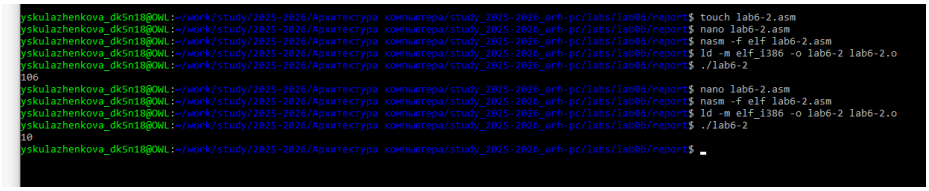

Рисунок 4.9: Запуск файла

Теперь заменим функцию iprintLF на print (рис. 4.10).

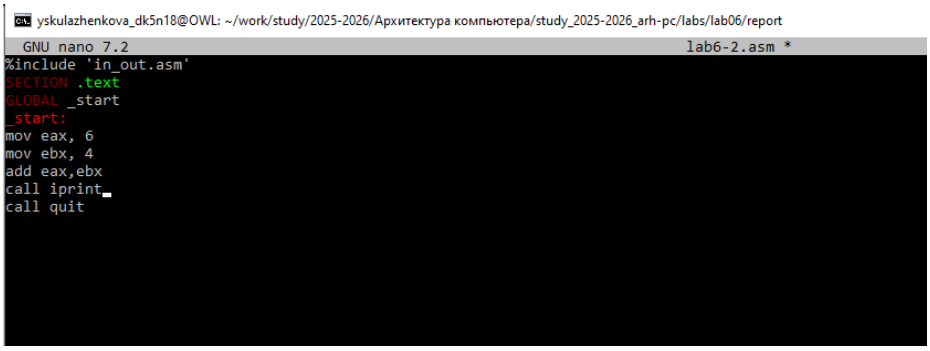

Рисунок 4.10: Новый код

Создадим исполняемый файл и запустим его (рис. 4.11).

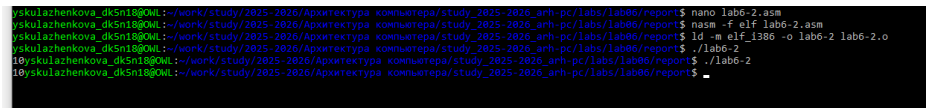

Рисунок 4.11: Работа файла

4.2 Выполнение арифметических операций в NASM

Напишем программу для вычисления значения арифметического выражения. Для этого создадим файл lab6-3.asm. Запишем нужный код программы (рис. 4.12).

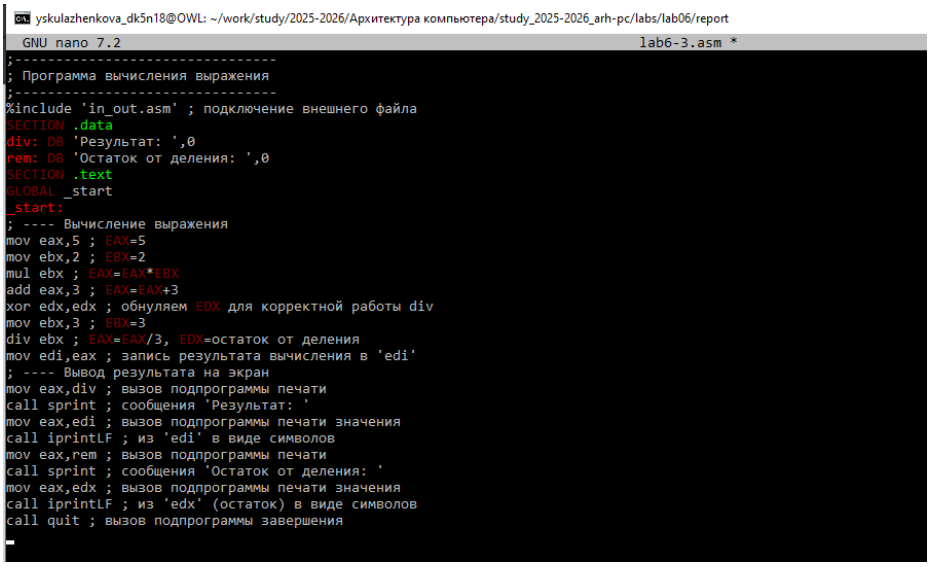

Рисунок 4.12: Код программы

Создадим исполняемый файл и проверим результат работы программы (рис. 4.13).

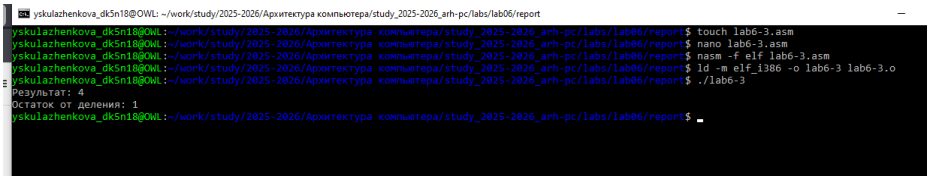

Рисунок 4.13: Выполнение программы

Далее найдём значение другого выражения. Для этого изменим текст программы. Проверим работу программы (рис. 4.14).

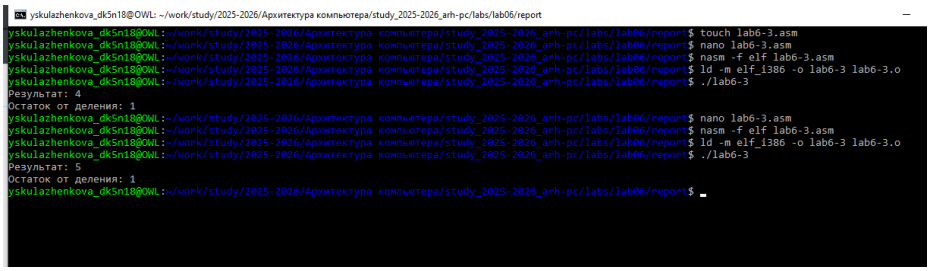

Рисунок 4.14: Исполнение файла

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. Для написания кода создадим файл variant.asm. Введем в файл variant.asm следующую программу (рис. 4.16).

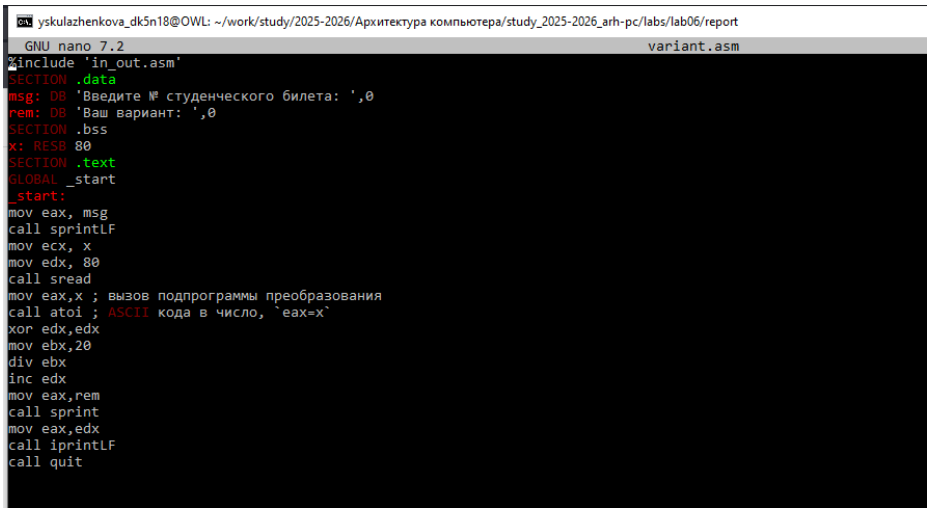

Рисунок 4.15: Редактирование файла

После этого проверим работу программы (рис. 4.16).

Рисунок 4.16: Выполнение файла

4.3 Вопросы лабораторной работы

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения „Ваш вариант:“?

Ответ: Строки, отвечающие за вывод сообщения „Ваш вариант:“

```
mov eax,rem
call sprint
```

2. Для чего используется следующие инструкции?

```
mov
ecx, x
mov
edx, 80
call sread
```

Ответ: данные инструкции используются для помещения адреса переменной x в регистр ecx, указания максимальной длины ввода (80 символов) и вызова подпрограммы чтения строки с клавиатуры

3. Для чего используется инструкция „call atoi“?

Ответ: функция atoi преобразует ascii-код символа в целое число и записывает результат в регистр eax.

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

Ответ: строки, отвечающие за вычисления варианта.

```
xor edx,edx
mov ebx,20
div ebx
inc edx
```

5. В какой регистр записывается остаток от деления при выполнении инструкции „div ebx“?

Ответ: остаток от деления записывается в регистр EDX.

6. Для чего используется инструкция „inc edx“?

Ответ: инструкция «inc edx» используется для увеличения на 1 остатка от деления (номера варианта), поскольку варианты нумеруются с 1, а не с 0.

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

Ответ: строки, отвечающие за вывод результата вычислений.

```
mov eax,edx
call iprintLF
```

5 Задание для самостоятельной работы

Напишем программу для вычисления значения функции. Затем проверим работу программы для различных функций и входных данных (рис. 5.1).

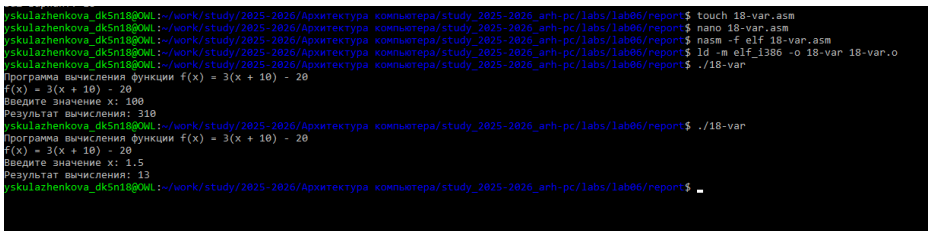

Рисунок 5.1: Самостоятельная работа

6 Выводы

В результате выполнения лабораторной работы была достигнута основная цель - освоение арифметических инструкций языка ассемблера NASM. На практике были успешно применены команды сложения (add), вычитания (sub), умножения (mul/imul) и деления (div/idiv), а также изучены особенности работы с операндами в регистрах и памяти. Особое внимание было уделено правилам использования регистров при операциях умножения и деления, где результат занимает фиксированные пары регистров в зависимости от размера операндов.

Важным результатом работы стало освоение методики преобразования данных между символьным и числовым представлением с использованием подпрограмм iprint, iprintLF и atoi. Практическое применение этих функций позволило организовать корректный ввод-вывод данных и выполнение арифметических операций над числами, введенными пользователем. Полученные навыки программирования на языке ассемблера NASM являются фундаментальными для дальнейшего изучения архитектуры ЭВМ и низкоуровневого программирования.