Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Авдеенко Марьяна Дмитриевна

Содержание

# 1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

# 2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

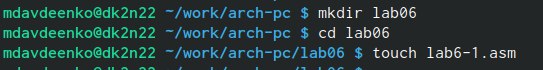
# 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес опе- ранда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации.

Существует три основных способа 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 ко- манде, Например: mov ax,2. \* Адресация памяти – операнд задает адрес в памяти. В команде указывается символи- ческое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

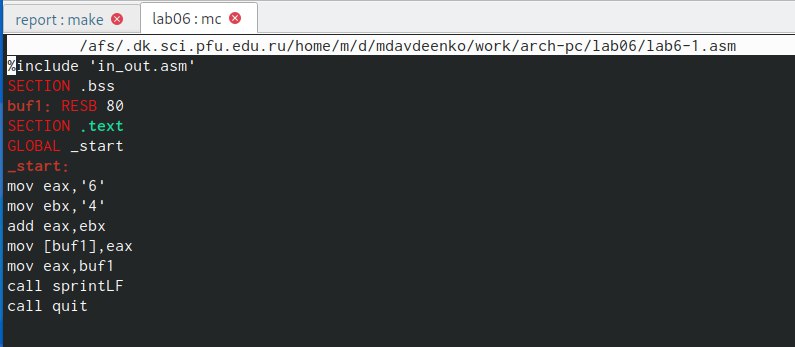
# 4 Выполнение лабораторной работы: символьные и численные данные в NASM

1. Создала каталог для программам лабораторной работы 6, перешла в него и создала файл lab6-1.asm (рис. ??).



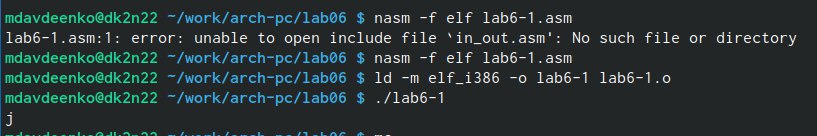
Создание каталога

1. Ввела в файл lab6-1.asm текст программы из листинга 6.1. (рис. ??).



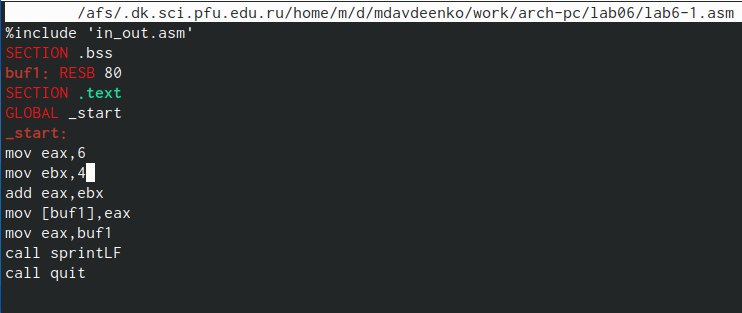
Программа вывода значения регистра eax

1. Создала исполняемый файл и запустила его, в данном случае при выводе значения регистра eax я ожидала увидеть число 10, но результатом стал символ j (рис. ??).



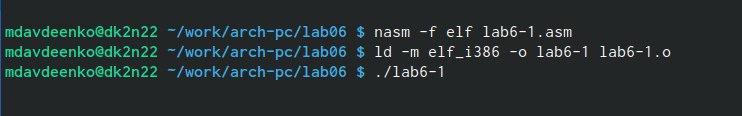
Запуск исполняемого файла

1. Далее изменила текст программы и вместо символов, записала в регистры числа. Ис- правила текст программы (Листинг 6.1), так как указано в задании (рис. ??).



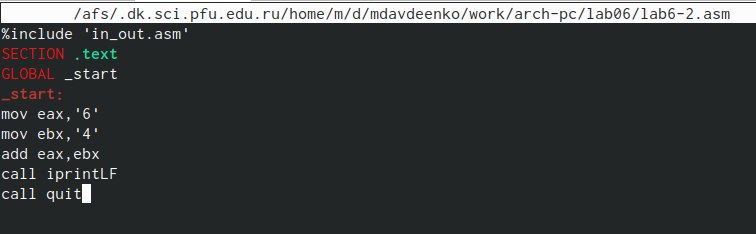
Замена строк программы

1. Создала исполняемый файл и запустила его (рис. ??).



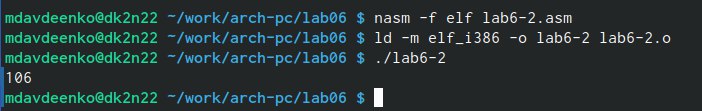
Проверка файла

1. Преобразовала текст программы из Листинга 6.1 с использованием функций в файле in\_out.asm: создала файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввела в него текст программы из листинга 6.2 (рис. ??).



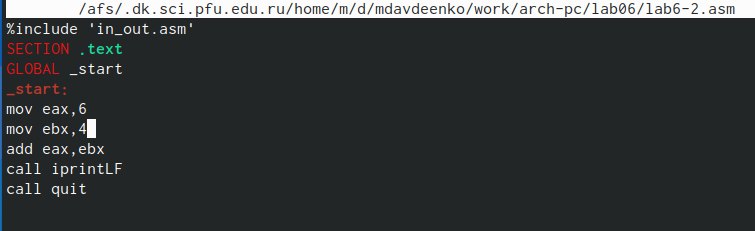
Преобразование файла с приминеием функций

1. Создала исполняемый файл и запустила его, в результате работы программы я получила число 106 (рис. ??).



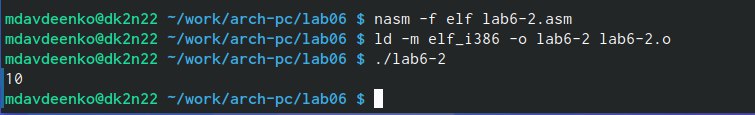
Исполняемый файл с числом

1. Аналогично предыдущему примеру изменила символы на числа, заменив строки так, как указано в задании (рис. ??).



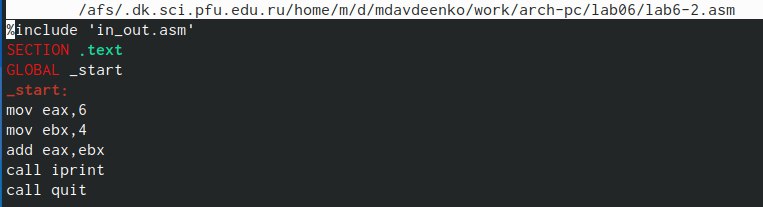
Исполняемый файл с числом

1. Создала исполняемый файл и запустила его (рис. ??).

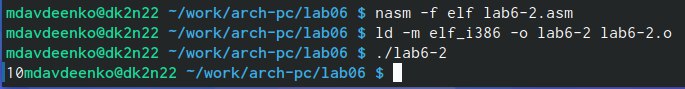


Исполняемый файл с числом

1. Заменила функцию iprintLF на iprint. Создала исполняемый файл и запустила его. Различие функций звключается в аргументе функции -LF, которая задает написание результата с переносом на новую строку (рис. ??, ??).



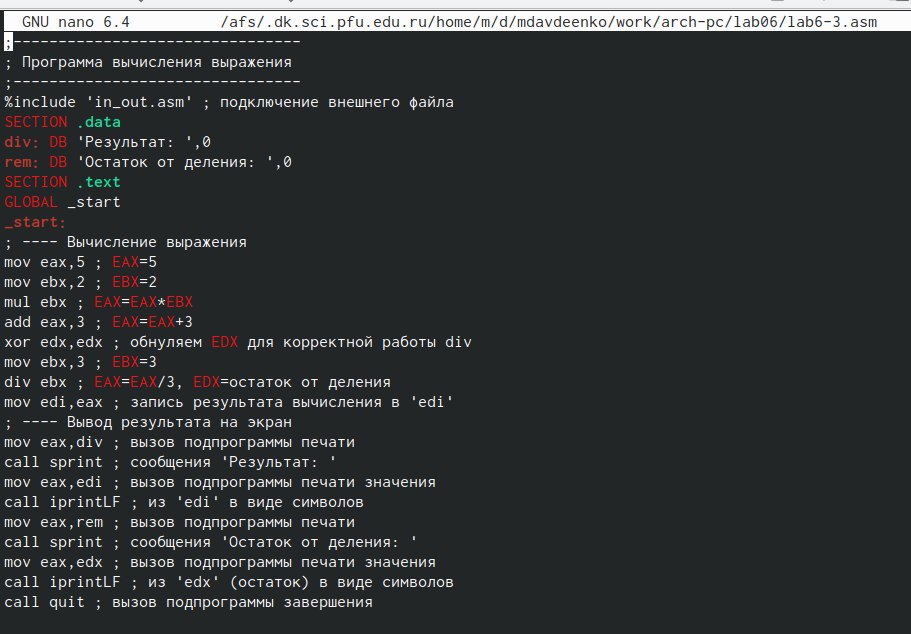
Замена функции iprintLF на iprint



Выведение числа измененной функцией

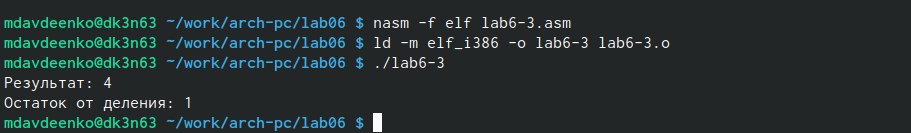
# 5 Выполнение лабораторной работы: выполнение арифметических операций в NASM

1. В качестве примера выполнения арифметических операций в NASM приведела программу вычисления арифметического выражения f(x) = (5 ∗ 2 + 3)/3. Создала файл lab6-3.asm в каталоге ~/work/arch-pc/lab06.
2. Изучила текст программы из листинга 6.3 и ввела его в lab6-3.asm (рис. ??).



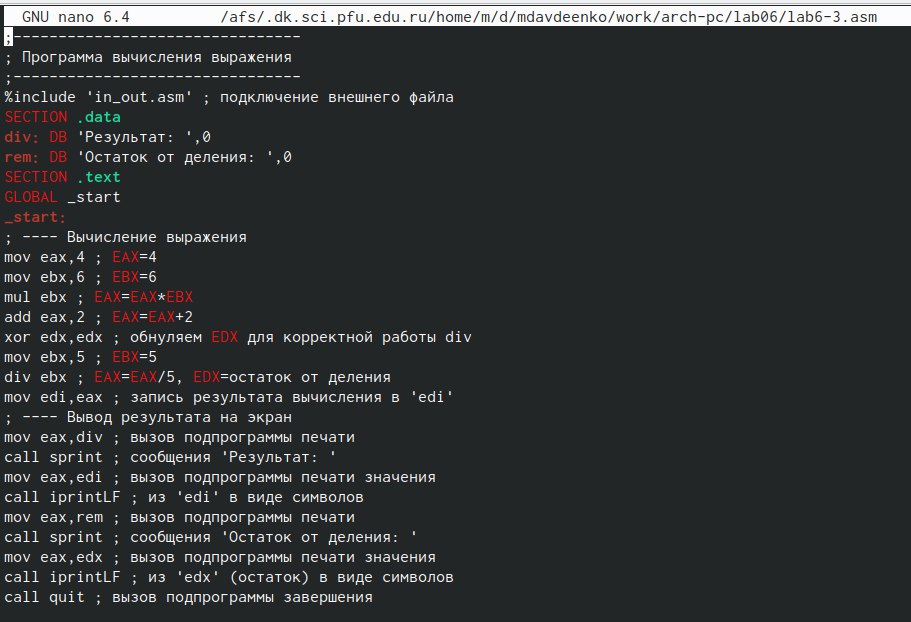
Применение листинга 6.3

1. Создала исполняемый файл и запустила его (рис. ??).



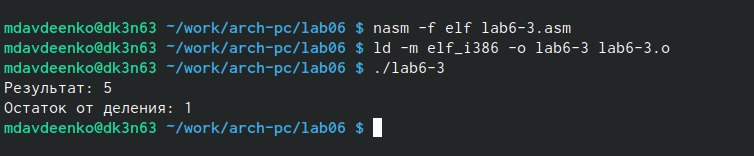
Проверка файла

1. Изменила текст программы для вычисления выражения f(x) = (4 ∗ 6 + 2)/5 (рис. ??).



Изменение программы

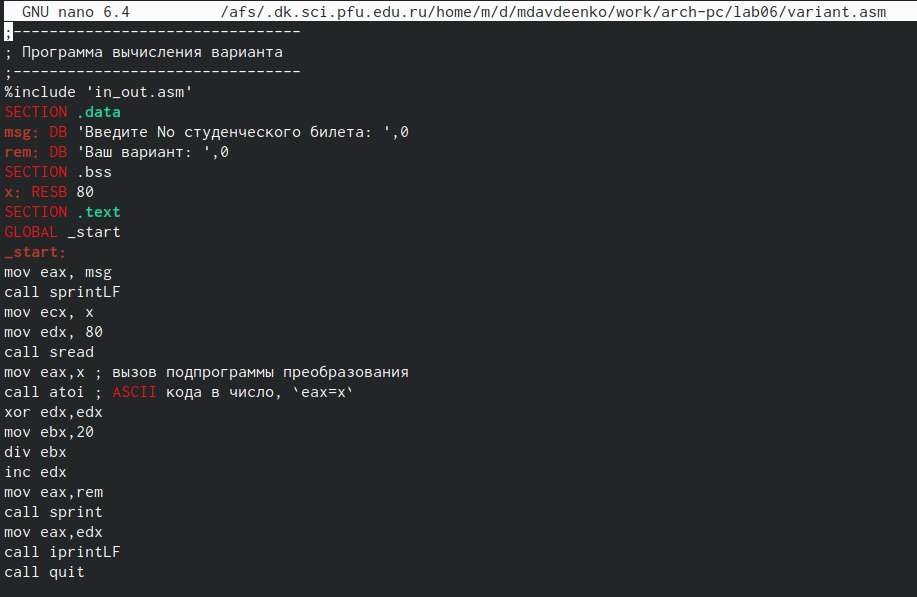
1. Создала исполняемый файл и проверила его работу (рис. ??).



Проверка файла

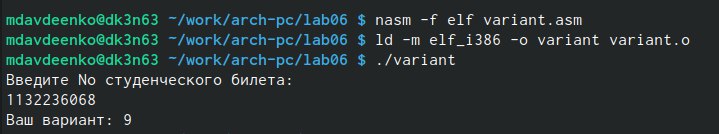
1. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

* вывести запрос на введение № студенческого билета
* вычислить номер варианта по формуле: (Sn mod 20) + 1, где Sn – номер студенческого билета (В данном случае a mod b – это остаток от деления a на b).
* вывести на экран номер варианта. Для этогно создала файл variant.asm в каталоге ~/work/arch-pc/lab06, внимательно изучила текст программы из листинга 6.4 и ввела в файл variant.asm (рис. ??).



Применение листинга 6.4

1. Создала исполняемый файл и запустила его (рис. ??).



Проверка файла

# 6 Оьветы на вопоросы

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

mov eax,rem call sprint

1. Инструкции mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx; mov edx,80 - для записи в регистр edx длины вводимой строки; call sread - вызова подпрограммы из внешнего файла, обеспесивающего ввод сообщения с клавиатуры.
2. Call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax.
3. За вычисление варианта отвечают строки:

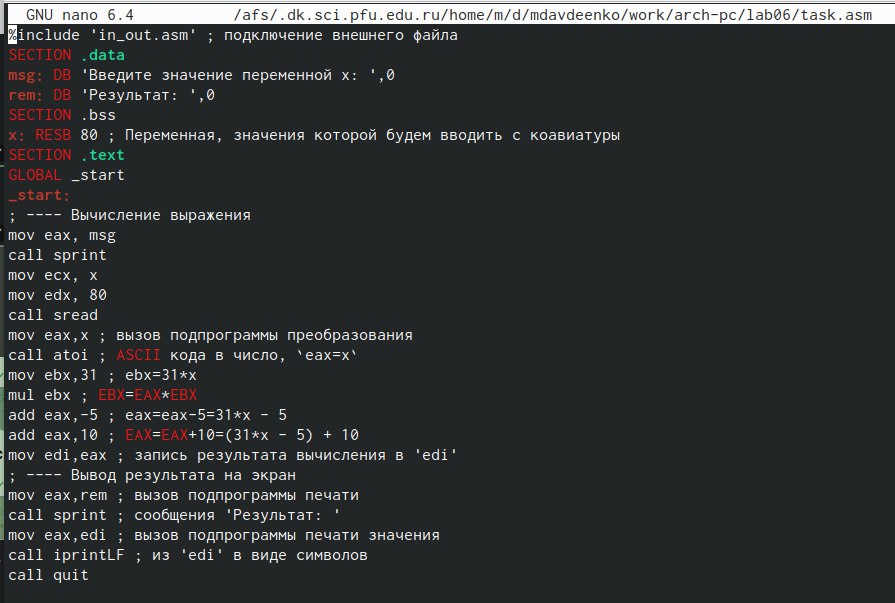
xor edx,edx ; обнуляем EDX для корректной работы div mov ebx,20 ; EBX=20 div ebx ; EAX=EAX/20, EDX=остаток от деления inc edx ; EDX=EDX+1

1. При выполнении инструкции div ebx остаток от деления записывается в регистр edx.
2. Инструкция inc edx увеличивает значение регистра edx на 1.
3. За вывод на экран результатов вычислений отвечают строки:

mov eax,edx call iprintLF

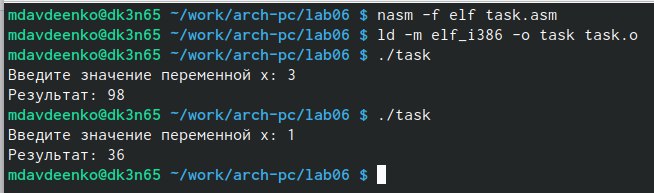
# 7 Выполнение заданий самостостоятельной работы

1. Создала файл task.asm, открюваю файл и ввожу в него текст программы для функции в 9 варианте (вариант, который был расчитан на предыдущих шагах) (рис. ??).



Программа файла task.asm

1. Создала исполняемый файл и проверила его работу для значений *x*=3 и *x*=1, программа вывела числовые значени, соответственно онв работает верно (рис. ??).



Проверка файла task.asm

# 8 Выводы

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

# Список литературы