Лабораторная работа №7

Арифметические операции в NASM.

Татьяна Алексеевна Коннова, НПИбд-01-22

Содержание

[1 Цель работы 1](#__RefHeading___Toc420_3008454844)

[1.1 Выполнение лабораторной работы 1](#__RefHeading___Toc422_3008454844)

[2 7.3.2. Выполнение арифметических операций в NASM 7](#__RefHeading___Toc424_3008454844)

[3 Самостоятельная работа 9](#__RefHeading___Toc426_3008454844)

[4 Выводы 11](#__RefHeading___Toc428_3008454844)

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM. ## Задание Выполнить операции с помощью ассемблера

## 1.1 Выполнение лабораторной работы

1. Создаем каталог для программам лабораторной работы No 7, перейдем в него и создаем файл lab7-1.asm: mkdir ~/work/arch-pc/lab07 cd ~/work/arch-pc/lab07 touch lab7-1.asm

(рис. 1)

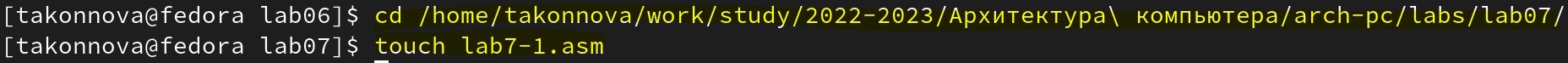


Рис. 1: touch

1. Рассмотрим примеры программ вывода символьных и численных значе- ний. Программы будут выводить значения записанные в регистр eax. Введите в файл lab7-1.asm текст программы из листинга 7.1. В данной про- грамме в регистр eax записывается символ 6 (mov eax,‘6’), в регистр ebx символ 4 (mov ebx,‘4’). Далее к значению в регистре eax прибавляем зна- чение 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. 2)



Рис. 2: lab7-1

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 свою очередь является кодом символа j

(рис. 3)

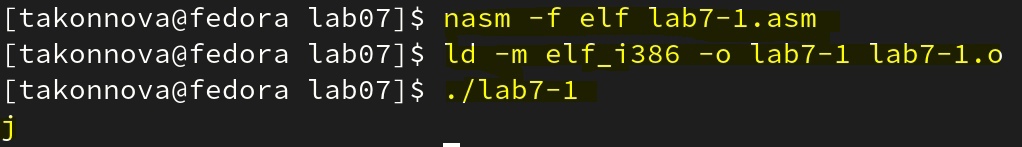


Рис. 3: Output j

1. Далее изменим текст программы и вместо символов, запишем в реги- стры числа. Исправим текст программы (Листинг 1) следующим образом:

mov eax,‘6’ mov ebx,‘4’ на строки mov eax,6 mov ebx,4

Как и в предыдущем случае при исполнении программы мы не получим число

(рис. 4)

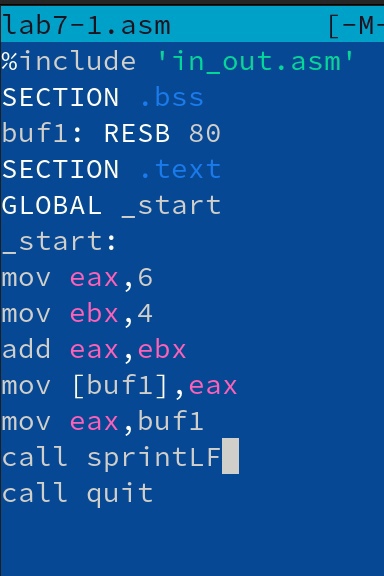


Рис. 4: Output

(рис. 5)

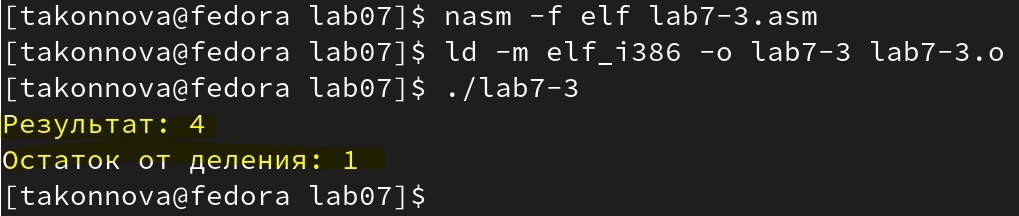


Рис. 5: Output\_prog

1. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определим какому символу соответствует код 10.

line feed, LF — «подача бумаги на строку») — управляющий символ ASCII ( 0x0A , 10 в десятичной системе счисления, при выводе которого курсор перемещается на следующую строку.

(рис. 6)

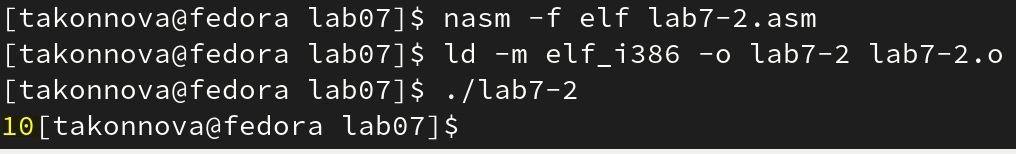


Рис. 6: Output

(рис. 7)

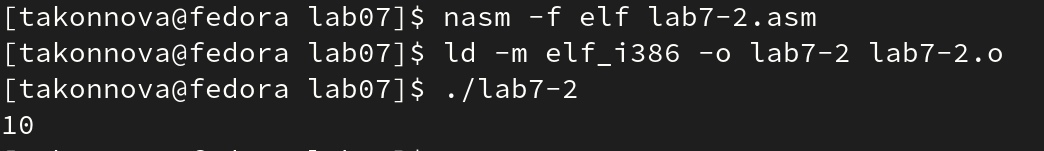


Рис. 7: Output

1. Как отмечалось выше, для работы с числами в файле in\_out.asm реализова- ны подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 7.1 с использованием этих функций. Создаем файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и введим в него текст программы из листинга 7.2. touch ~/work/arch-pc/lab07/lab7-2.asm

Создаем исполняемый файл и запускаем 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 7.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

(рис. 8)

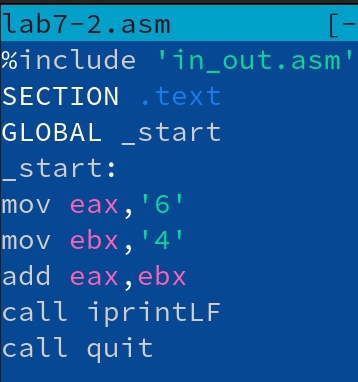


Рис. 8: LAB7-2

(рис. 9)

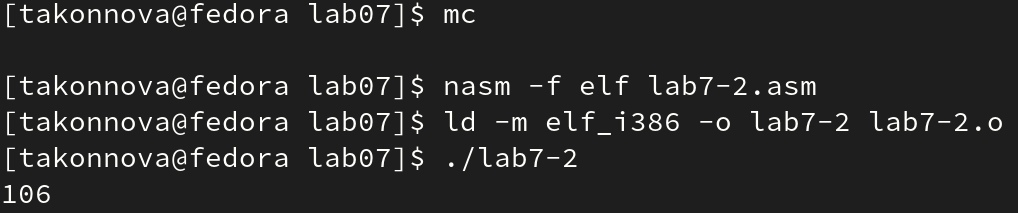


Рис. 9: Output

1. Аналогично предыдущему примеру изменим символы на числа. Заменим строки

mov eax,‘6’ mov ebx,‘4’ на строки mov eax,6 mov ebx,4

Создаем исполняемый файл и запускаем его. Заменяем функцию iprintLF на iprint. Создаем исполняемый файл и запускаем его.

(рис. 10)

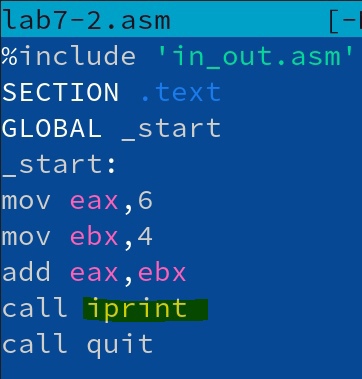


Рис. 10: Iprint

# 2 7.3.2. Выполнение арифметических операций в NASM

1. В качестве примера выполнения арифметических операций в NASM при- ведем программу вычисления арифметического выражения f(x) = (5 \* 2 + 3)/3. Создаем файл lab7-3.asm в каталоге ~/work/arch-pc/lab07: touch ~/work/arch-pc/lab07/lab7-3.asm Внимательно изучаем текст программы из листинга 7.3 и вводим в lab7-3.asm.

(рис. 11)

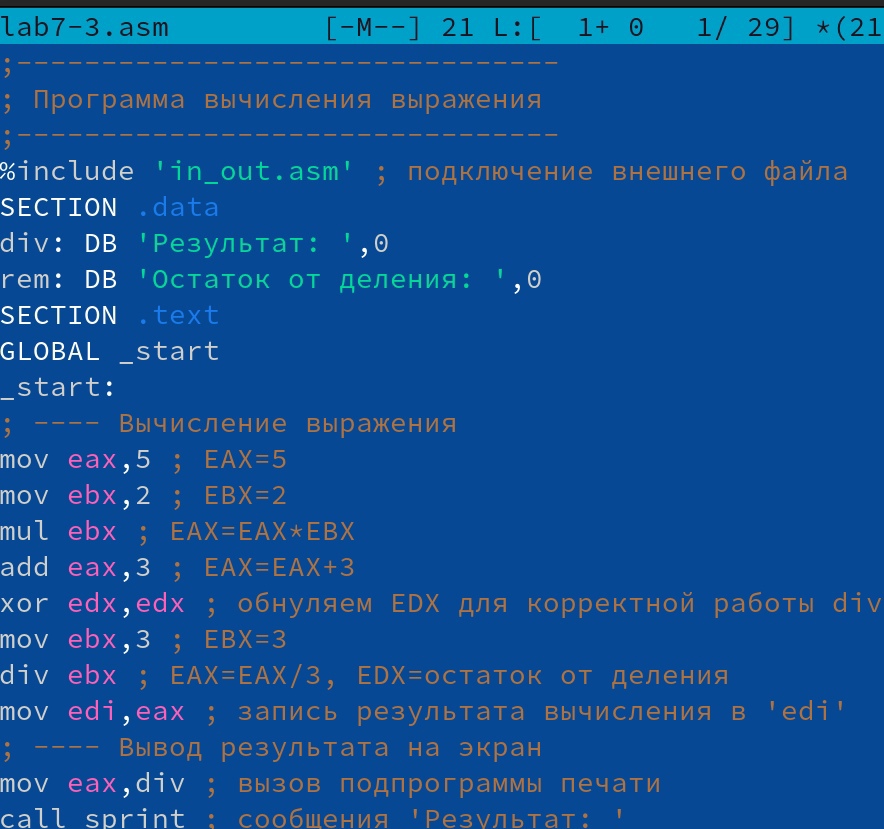


Рис. 11: Experession

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

mov eax, rem

call sprint

1. Для чего используется следующие инструкции? nasm mov ecx, x mov edx, 80 call sread Подобное необходимо затем, чтобы мы ввели посыл не более 80 символьных значений в переменную х
2. Для чего используется инструкция “call atoi”? Чтобы работать не с символами из ASCII работающей по порядковым номерам, а с числами, то есть с реальными цифрами.
3. Какие строки листинга 7.4 отвечают за вычисления варианта?

xor edx,edx

mov ebx,20

div ebx

inc edx

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

edx

1. Для чего используется инструкция “inc edx”? Чтобы провести требуемую операцию по нахождению остатка от 20 и прибавления единицы. В частности, последнего.
2. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений?

mov eax,edx call iprintLF

# 3 Самостоятельная работа

Выполним самостоятельное задание, следуя инструкции. Произведем вычисления 19 функции, воспользовавшись ассемблером.

(рис. 12)

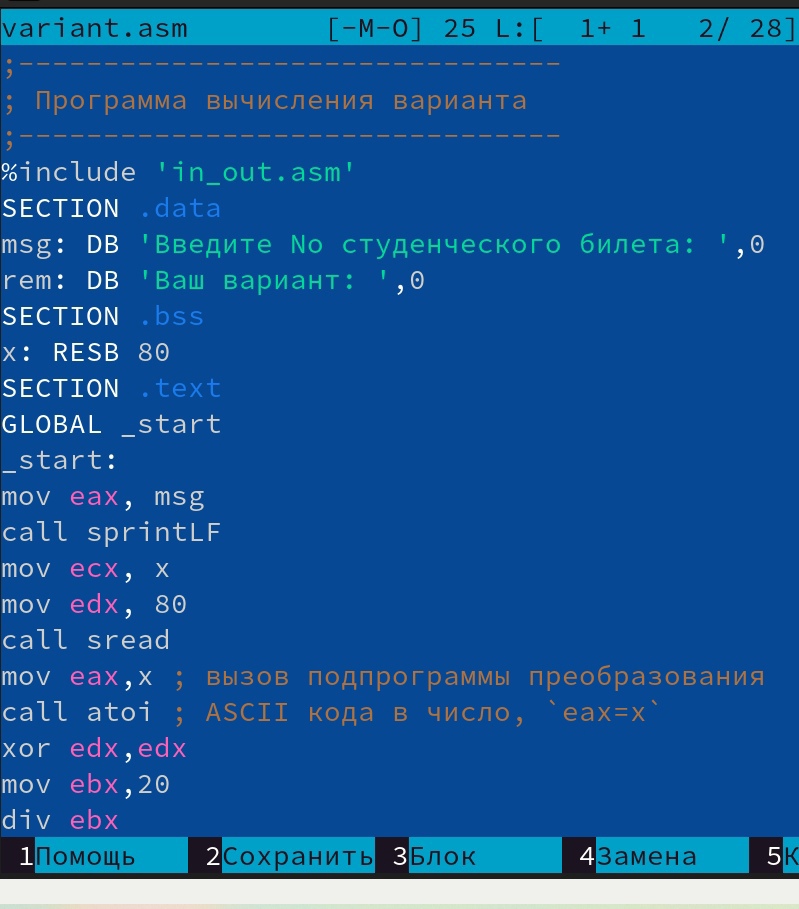


Рис. 12: Variant

(рис. 13)



Рис. 13: Zadanie

# 4 Выводы

Освоение арифметических инструкций языка ассемблера NASM.