Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Кузьмина Мария Константиновна

Содержание

# 1 Цель 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 Выполнение лабораторной работы

## 3.1 Символьные и численные данные в NASM

С помощью mkdir создаем директорию для создания файлов лабораторной работы, переходим в созданный каталог(рис. 1):



Рис. 1: снимок экрана

С помощью утилиты touch создаем файл lab6-1.asm, копируем в текущий каталог файл in\_out.asm и открываем созданный файл; вставляем в него программу вывода значения регистра eax (рис. 2):

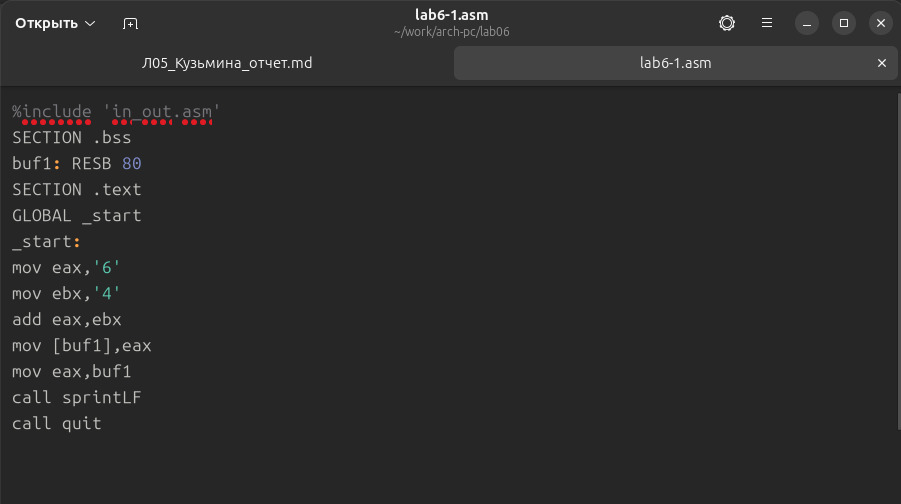


Рис. 2: снимок экрана

Сохраняем исполняемый файл программы и запускаем его (рис. 3):

|  |
| --- |
| Рис. 3: снимок экрана |

Рис. 3: снимок экрана

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

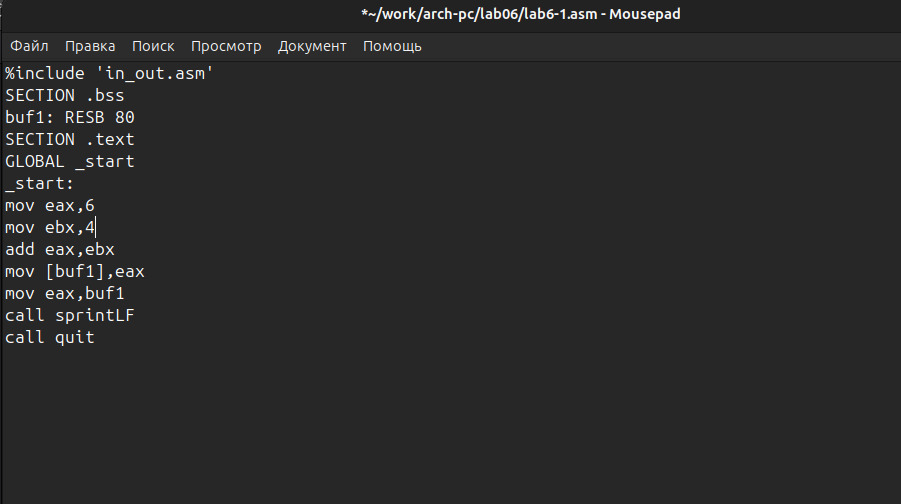


Рис. 4: снимок экрана

Создаем новый исполняемый файл программы и запускаем его. При исполнении программы мы не получаем число 10. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определяем, что 10 соответствует символу перевода строки. (рис. 5):

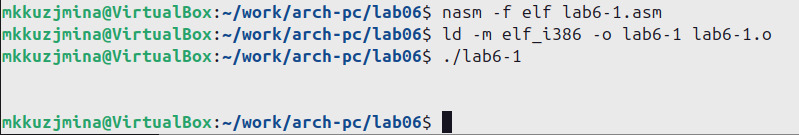


Рис. 5: снимок экрана

С помощью touch создаем новый файл lab6-2.asm, открываем mousepad и вводим текст программы для вывода значения регистра eax (рис. 6):

Рис. 6: снимок экрана

Рис. 6: снимок экрана

(рис. 7):



Рис. 7: снимок экрана

Создаем и запускаем исполняемый файл lab6-2. На выxоде получаем число 106, потому что программа позволяет вывести число. (рис. 8):

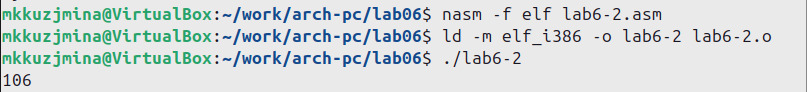


Рис. 8: снимок экрана

Заменяем в тексте программы файла lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 9):

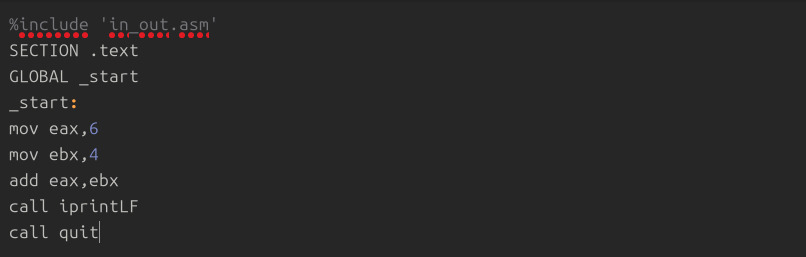


Рис. 9: снимок экрана

Создаем и запускаем новый исполняемый файл. Программа складывает непосредственно сами числа, поэтому вывод 10.

(рис. 10):

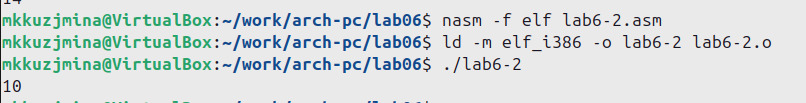


Рис. 10: снимок экрана

Заменяем в тексте программы функцию iprintLF на iprint (рис. 11):

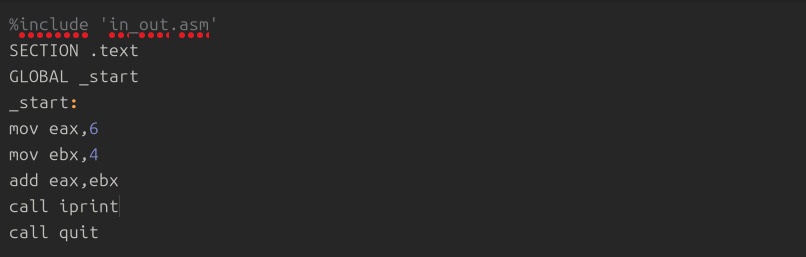


Рис. 11: снимок экрана

Создаем и запускаем новый исполняемый файл. Вывод изменился, потому что iprintLF добавил перенос строки, а iprint нет. (рис. 12):

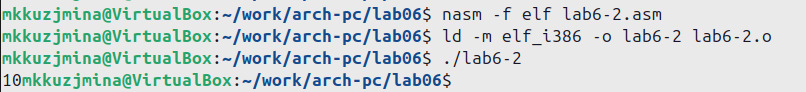


Рис. 12: снимок экрана

## 3.2 Выполнение арифметических операций в NASM

Создаем файл lab6-3.asm с помощью touch (рис. 13):

Рис. 13: снимок экрана

Рис. 13: снимок экрана

Вводим в созданный файл текст программы для вычисления значения данного выражения (рис. 14):

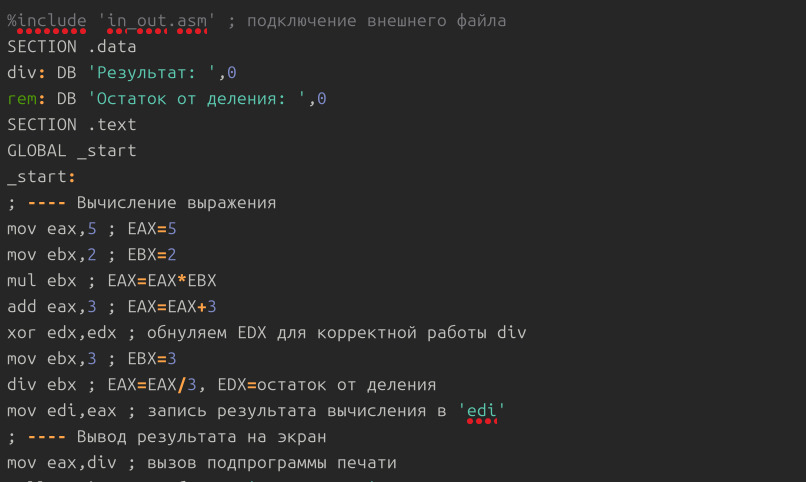


Рис. 14: снимок экрана

Создаем и запускаем файл (рис. 15):

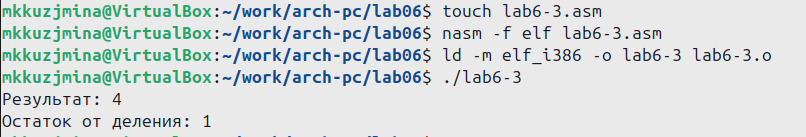


Рис. 15: снимок экрана

Изменяем текст, чтобы программа вычисляла значение нового выражения (4 \* 6 + 2)/5 (рис. 16):

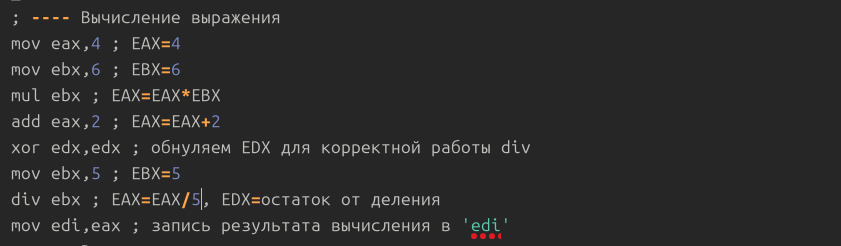


Рис. 16: снимок экрана

%include 'in\_out.asm' ; подключение внешнего файла  
SECTION .data  
div: DB 'Результат: ',0  
rem: DB 'Остаток от деления: ',0  
SECTION .text  
GLOBAL \_start  
\_start:  
; ---- Вычисление выражения  
mov eax,4 ; EAX=4  
mov ebx,6 ; EBX=6  
mul ebx ; EAX=EAX\*EBX  
add eax,2 ; EAX=EAX+2  
xor edx,edx ; обнуляем EDX для корректной работы div  
mov ebx,5 ; EBX=5  
div ebx ; EAX=EAX/5, EDX=остаток от деления  
mov edi,eax ; запись результата вычисления в 'edi'  
; ---- Вывод результата на экран  
mov eax,div ; вызов подпрограммы печати  
call sprint ; сообщения 'Результат: '  
mov eax,edi ; вызов подпрограммы печати значения  
call iprintLF ; из 'edi' в виде символов  
mov eax,rem ; вызов подпрограммы печати  
call sprint ; сообщения 'Остаток от деления: '  
mov eax,edx ; вызов подпрограммы печати значения  
call iprintLF ; из 'edx' (остаток) в виде символов  
call quit ; вызов подпрограммы завершения

Создаем и запускаем файл (рис. 17):

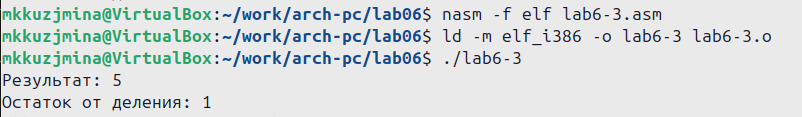


Рис. 17: снимок экрана

Создаем файл variant.asm, вводим в файл текст программы для вычисления варианта по номеру студенческого билета(рис. 18):

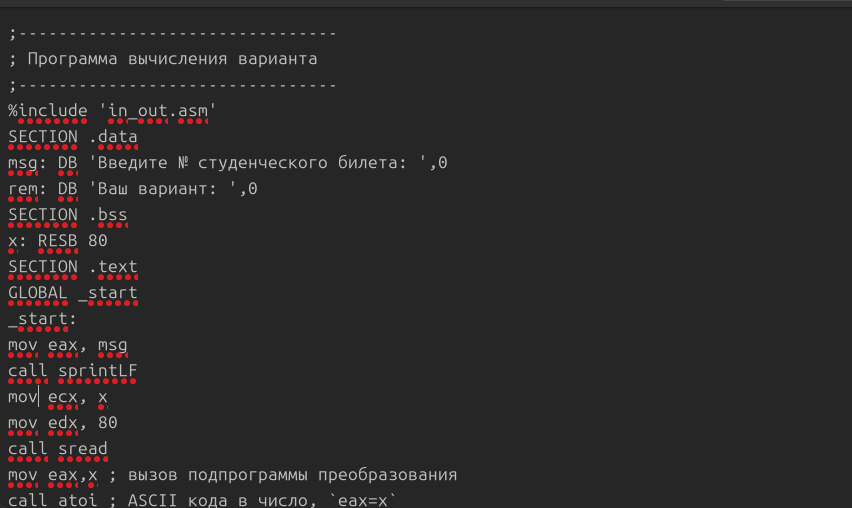


Рис. 18: снимок экрана

Создаем и запускаем исполняемый файл. Вводим номер своего студенческого билета. (рис. 19):

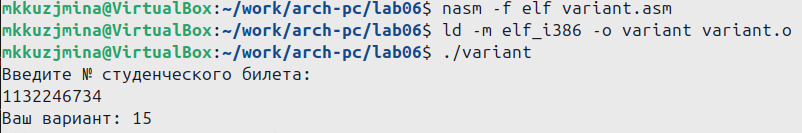


Рис. 19: снимок экрана

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? Ответ:

mov eax,rem  
call sprint

1. Для чего используется следующие инструкции? mov ecx, x mov edx, 80 call sread

Ответ: mov ecx, x — загружает адрес буфера, куда будет сохранён ввод. mov edx, 80 — указывает максимальное количество символов для ввода (80). call sread — вызывает функцию для чтения строки с клавиатуры.

1. Для чего используется инструкция “call atoi”? Ответ: преобразует введённую строку в целое число.
2. Какие строки листинга 6.4 отвечают за вычисления варианта? Ответ:

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

1. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”? Ответ: остаток от деления записывается в регистр edx
2. Для чего используется инструкция “inc edx”? Ответ: увеличивает значение, находящееся в регистре edx, на 1.
3. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений? Ответ:

mov eax,edx  
call iprintLF

## 3.3 Выполнение заданий для самостоятельной работы

Создаю и открываем файл lab6-4.asm , вводим в него текст программы для вычисления значения (5 + x)^2 − 3, (вариант 15) (рис. 20):

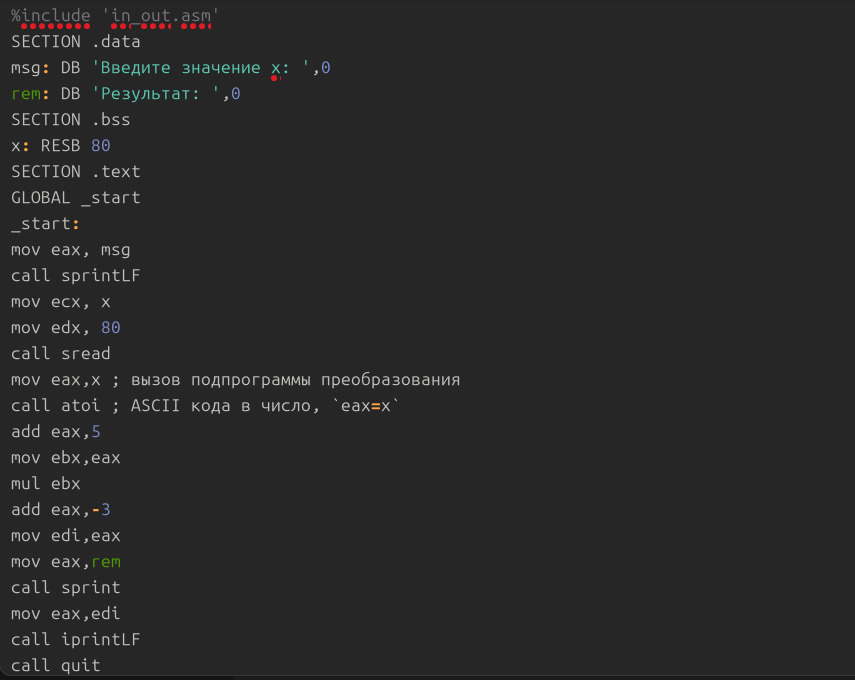


Рис. 20: снимок экрана

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите значение x: ',0  
rem: DB 'Результат: ',0  
SECTION .bss  
x: RESB 80  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax, msg  
call sprintLF  
mov ecx, x  
mov edx, 80  
call sread  
mov eax,x ; вызов подпрограммы преобразования  
call atoi ; ASCII кода в число, `eax=x`  
add eax,5  
mov ebx,eax  
mul ebx  
add eax,-3  
mov edi,eax  
mov eax,rem  
call sprint  
mov eax,edi  
call iprintLF  
call quit

Создаем и запускаем исполняемый файл, после вводим значения x1, x2 (рис. 21):

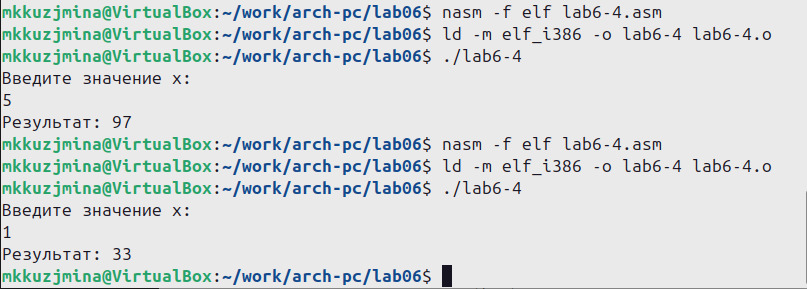


Рис. 21: снимок экрана

# 4 Выводы

При выполнении лабораторной работы были освоены арифметические инструкции языка ассемблера NASM.