Лабораторная работа №6.

Арифметические операции в NASM.

Селиванов Вячеслав Алексеевич

Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

Символьные и численные данные в NASM Выполнение арифметических операций в NASM Задания для самостоятельной работы

# 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

## 4.1 Символьные и численные данные в NASM

С помощью команды mkdir создаю новую директорию,в которой буду создавать файлы с программами во время всей лабораторной работы №6 (рис. [1](#fig:001)). Перехожу в созданный каталог

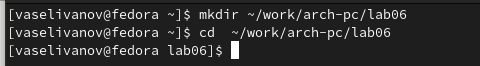


Figure 1: Создание каталога

Создаю файл lab6-1.asm,используя команду”touch” (рис. [2](#fig:002)).

Figure 2: Создание файла

Figure 2: Создание файла

Копирую в созданный каталог файл in\_out.asm,потому что он будет использоваться и в других программах (рис. [3](#fig:003)).

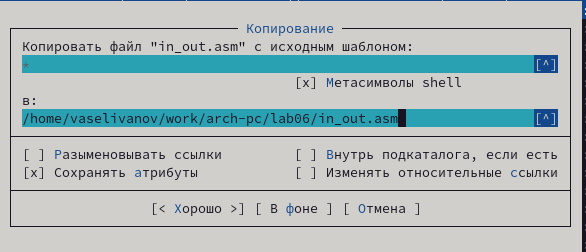


Figure 3: Копирование файла

Открываю файл lab6-1.asm в nano и вставляю в него программу ввода значения регистра eax (рис. [4](#fig:004)).

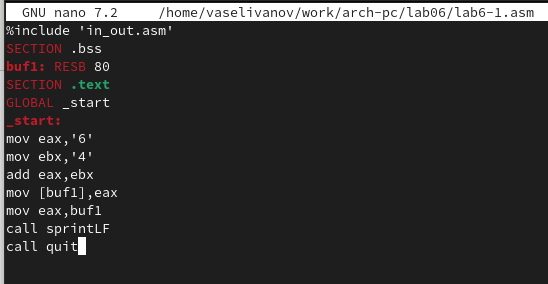


Figure 4: Редактирование файла

%include 'in\_out.asm'  
SECTION .bss  
buf1: RESB 80  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax,'6'  
mov ebx,'4'  
add eax,ebx  
mov [buf1],eax  
mov eax,buf1  
call sprintLF  
call quit

Создаю объектный файл и после его компановки запускаю программу (рис. [5](#fig:005)). Программа выводит символ “j”,потому что программа вывела символ,которые соответствует в системе ASCII сумме двоичных символов 4 и 6.

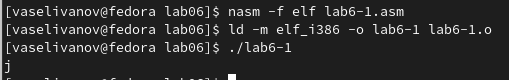


Figure 5: Запуск программы

Теперь изменяю в тексте программы символы ‘6’ и ‘4’ на цифры 6 и 4 (рис. [6](#fig:006)).

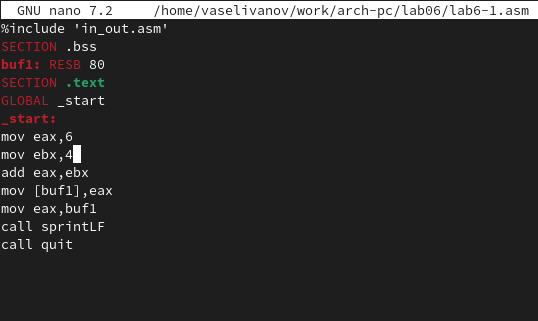


Figure 6: Редактирование программы

%include 'in\_out.asm'  
SECTION .bss  
buf1: RESB 80  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
mov [buf1],eax  
mov eax,buf1  
call sprintLF  
call quit

Создаю новый исполняемый файл и запускаю программу (рис. [7](#fig:007)). Теперь у меня выводится символ с кодом 10,это символ перевода строки, он не отображается при выводе на экран.

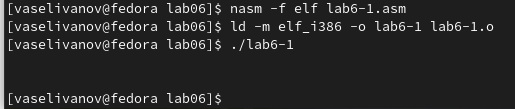


Figure 7: Запуск исполняемого файла

Создаю новый файл под названием lab6-2.asm и проверяю его наличие, используя ls (рис. [8](#fig:008)).

Figure 8: Создание нового файла

Figure 8: Создание нового файла

Ввожу в файл текст уже другой программы для вывода eax (рис. [9](#fig:009)).

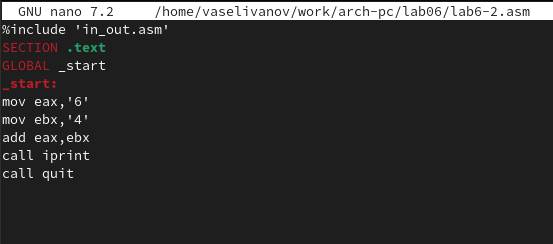


Figure 9: Редактирование программы

%include 'in\_out.asm'  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax,'6'  
mov ebx,'4'  
add eax,ebx  
call iprint  
call quit

Создаю и запускаю исполняемый файл lab6-2 (рис. [10](#fig:010)).Теперь программа выводит 106,потому что программа выводит именно число,не символ, хотя всё еще происходит сложение кодов символов “6” и “4”.

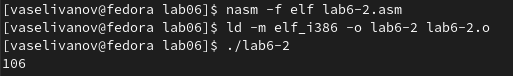


Figure 10: Запуск программы

Заменяю в текста данной программы символы ‘4’ и ‘6’ на числа 6 и 4 (рис. [11](#fig:011)).

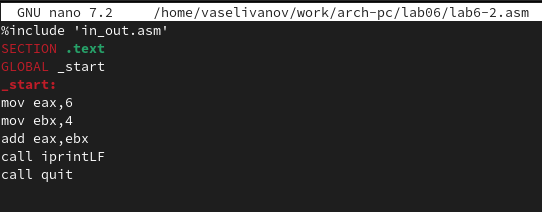


Figure 11: Редактирование программы

%include 'in\_out.asm'  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
call quit

Снова создаю исполняемый файл программы lab6-2 (рис. [12](#fig:012)). Теперь программа складывает именно числа,поэтому выводом является сумма 4+6,которая равна 10.

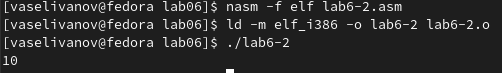


Figure 12: Запуск программы

Заменяю в тексте программы функцию iprintLF на iprint (рис. [13](#fig:013)).

Figure 13: Редактирование программы

Figure 13: Редактирование программы

Создаю и запускаю исполняемый файл (рис. [14](#fig:014)). После завершение программы меня не перебрасывает на следующую строку а оставляет на той же.

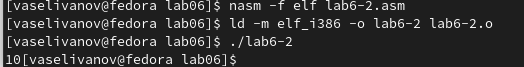


Figure 14: Запуск программы

## 4.2 Выполнение арифметических операций в NASM

Создаю файл,называю его lab6-3.asm, используя команду touch (рис. [15](#fig:015)).

Figure 15: Создание файла

Figure 15: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения f(x)=(5\*2+3)/3 (рис. [16](#fig:016)).

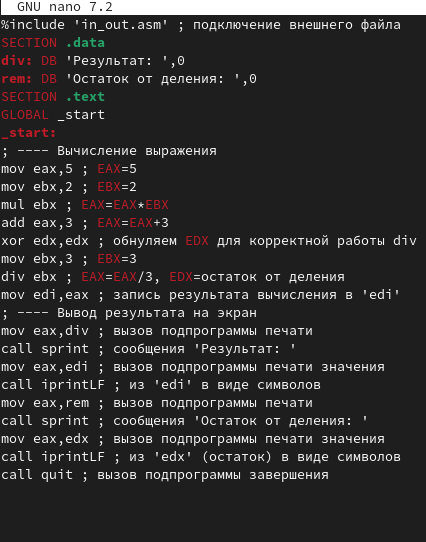


Figure 16: Редактирование программы

Запускаю созданный исполняемый файл (рис. [17](#fig:017)).

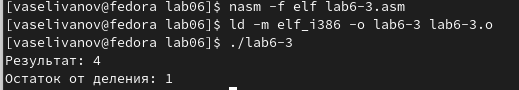


Figure 17: Запуск программы

Изменяю программу так,чтобы она вычисляла другое выражение-f(x)=(4\*6+2)/5 (рис. [18](#fig:018)).

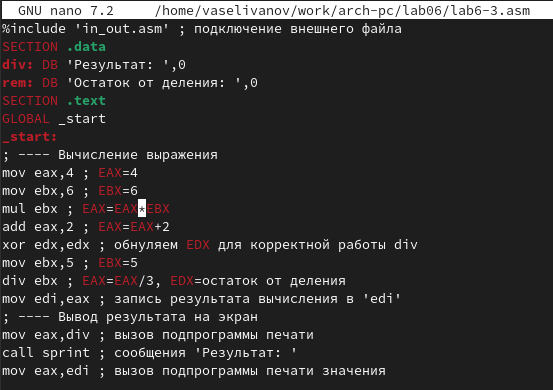


Figure 18: Редактирование программы

%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 завершени

Создаю исполняемый файл и запускаю программу (рис. [19](#fig:019)). Программа посчитала выражение абсолютно верно.

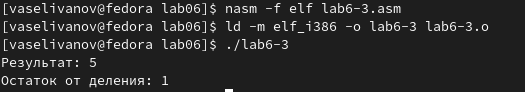


Figure 19: Запуск программы

Создаю новый файл variant.asm (рис. [20](#fig:020)).

Figure 20: Создание файла

Figure 20: Создание файла

Ввожу в файл текст программы,который вычисляет задания по номеру студенческого билета (рис. [21](#fig:021)).

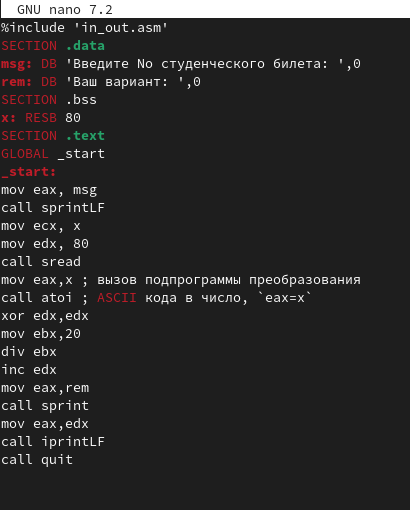


Figure 21: Редактирование программы

Создаю исполняемый файл и запускаю программу (рис. [22](#fig:022)). Ввожу номер своего студенческого. Программа вывела,что мой вариант№12.

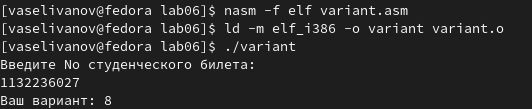


Figure 22: Запуск программы

### 4.2.1 Ответы на вопросы

1. За вывод сообщения “Ваш вариант” отвечают строчки кода

mov eax,rem  
call sprint

1. Инструкция mov ес, х используется, чтобы положить адрес вводимой строки х в регистр есх mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
2. За вывод на экран результатов вычислений отвечают строки:
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:

хот edx,edx ; обнуление еах 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

## 4.3 Задания для самостоятельной работы

Создаю файл lab6-4.asm,используя команду touch (рис. [23](#fig:023)).

Figure 23: Создание файла

Figure 23: Создание файла

Открываю файл, ввожу в него для вычисления выражения,которое мне выпало (№8) (11 + x) \* 2 - 6 (рис. [24](#fig:024)).

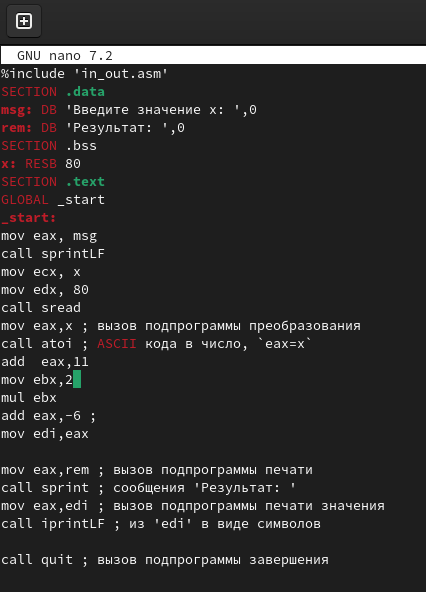


Figure 24: Редактирование программы

%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,11  
mov ebx,2   
mul ebx  
add eax,-6 ;  
mov edi,eax  
  
mov eax,rem ; вызов подпрограммы печати  
call sprint ; сообщения 'Результат: '  
mov eax,edi ; вызов подпрограммы печати значения  
call iprintLF ; из 'edi' в виде символов  
  
call quit ; вызов подпрограммы завершения

Создаю исполняемый файл и запускаю программу (рис. [25](#fig:025)). Программа выдает правильный ответ.

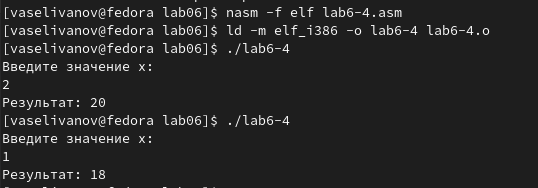


Figure 25: Запуск программы

# 5 Выводы

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

# Список литературы

1. Лабораторная работа №6
2. Таблица ASCII