Лабораторная работа №6

Арифметическиеоперации в NASM

Богданюк Анна Васильевна

Содержание

# Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# Задание

1. Выполнение лабораторной работы
2. Задние для самостоятельной работы

# Теоретическое введение

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 команде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Схема команды целочисленного сложения add (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда add работает как с числами со знаком, так и без знака.

Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: inc (от англ. increment) и dec (от англ. decrement), которые увеличивают и уменьшают на 1 свой операнд.

Команда neg рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.

Для деления, как и для умножения, существует 2 команды div (от англ. divide - деление) и idiv.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Расширенная таблица ASCII состоит из двух частей. Первая (символы с кодами 0-127) является универсальной (см. Приложение.), а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно. Для выполнения лабораторных работ в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно.

# Выполнение лабораторной работы

1. Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы №6, перехожу в него и создаю файл lab6-1.asm (рис. [1](#fig:001)).

Figure 1: Создание файла в каталоге для лабораторной работы

Figure 1: Создание файла в каталоге для лабораторной работы

Ввожу текст программы из листинга.В данной программе в регистр eax записывается символ 6 (mov eax,‘6’), в регистр ebx символ 4 (mov ebx,‘4’). Далее к значению в регистре eax прибавляем значение 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](#fig:002)).

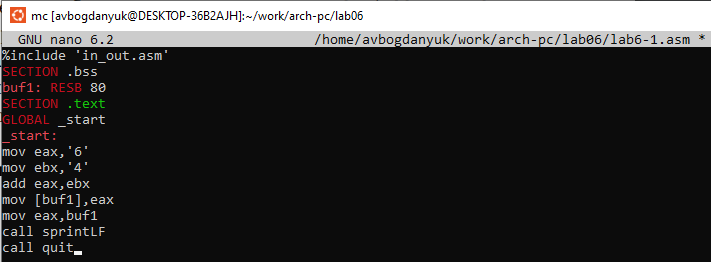


Figure 2: Файл lab6-1.asm

Листинг:

%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

Копирую файл in\_out.asm в тот же каталог, где находится lab6-1.asm (рис. [3](#fig:003)).

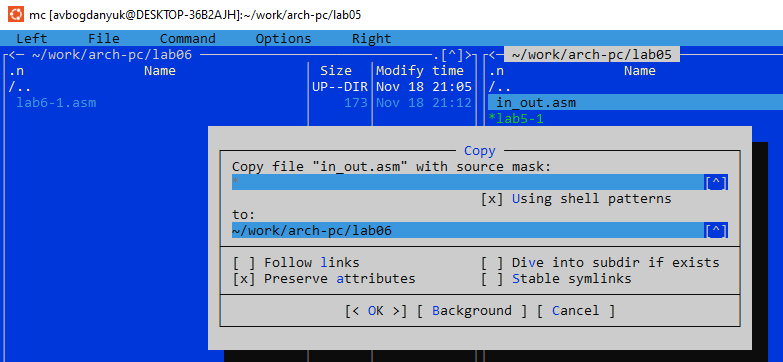


Figure 3: Копирование файла

Создаю исполняемый файл и запускаю его. 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. [4](#fig:004)).

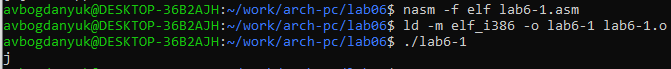


Figure 4: Создание и запуск

Изменяю текст программы и вместо символов, пишу в регистры числа (рис. [5](#fig:005)).

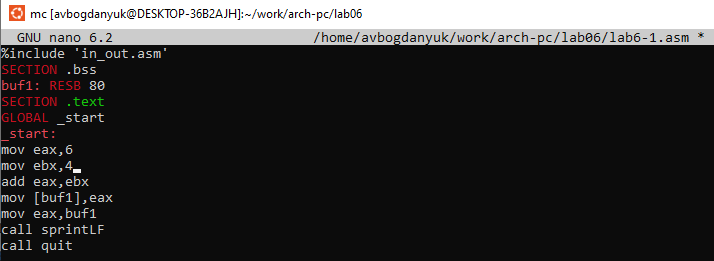


Figure 5: Файл lab6-1.asm

Создаю исполняемый файл и запускаю его. Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Это символ перевода строки, символ не отображается на экране (рис. [6](#fig:006)).

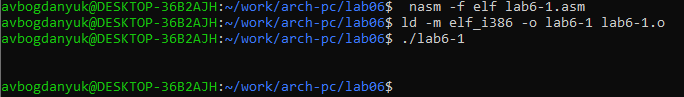


Figure 6: Создание и запуск

Создаю файл lab6-2.asm и ввожу в него текст программы листинга (рис. [7](#fig:007)).

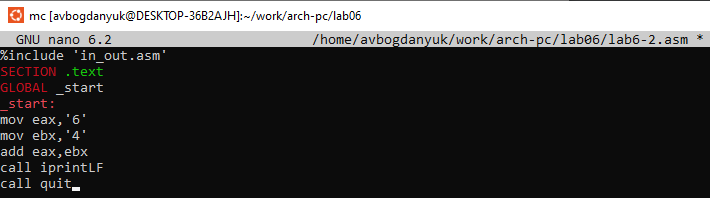


Figure 7: Файл lab6-2.asm

Листинг:

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

Создаю исполнительный файл и запускаю 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 6.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число (рис. [8](#fig:008)).

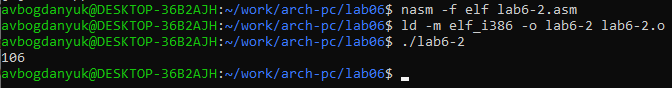


Figure 8: Создание и запуск

Измению символы на числа (рис. [9](#fig:009)).

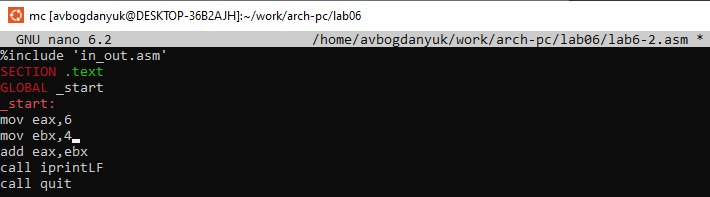


Figure 9: Файл lab6-2.asm

Создаю исполнительный файл и запускаю его. При исполнении программы выводится число 10, так как складываются числа 6 и 4 (рис. [10](#fig:010)).

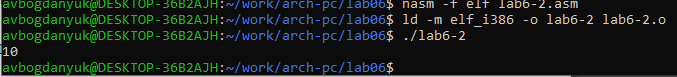


Figure 10: Файл lab6-2.asm

Заменяю функцию iprintLF на iprint. (рис. [11](#fig:011)).

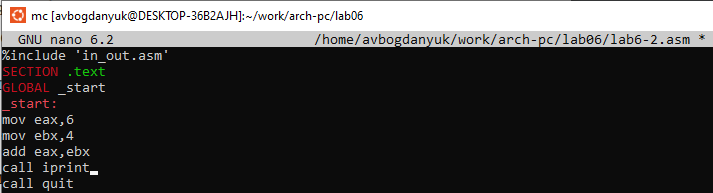


Figure 11: Файл lab6-2.asm

Создаю исполнительный файл и запускаю его. iprint не добавляет символ переноса строки (рис. [12](#fig:012)).

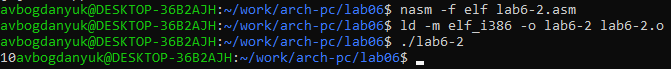


Figure 12: Создание и запуск

Создаю файл lab6-3.asm и ввожу в него текст программы листинга(рис. [13](#fig:013)).

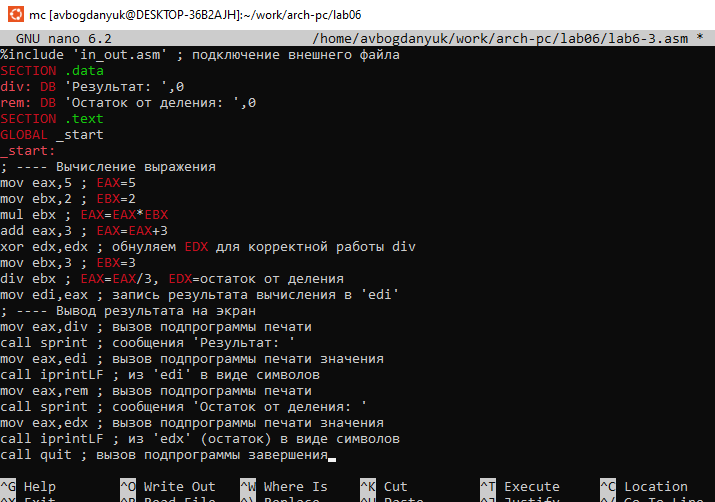


Figure 13: Файл lab6-3.asm

%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,5 ; EAX=5  
mov ebx,2 ; EBX=2  
mul ebx ; EAX=EAX\*EBX  
add eax,3 ; EAX=EAX+3  
xor edx,edx ; обнуляем EDX для корректной работы div  
mov ebx,3 ; EBX=3  
div ebx ; EAX=EAX/3, 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

Создаю исполнительный файл и запускаю его (рис. [14](#fig:014)).

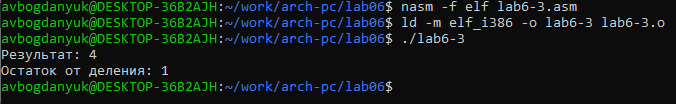


Figure 14: Создание и вывод

Изменяю текст программы для вычисления выражения f(x) = (4 \* 6 + 2)/5. (рис. [15](#fig:015)).

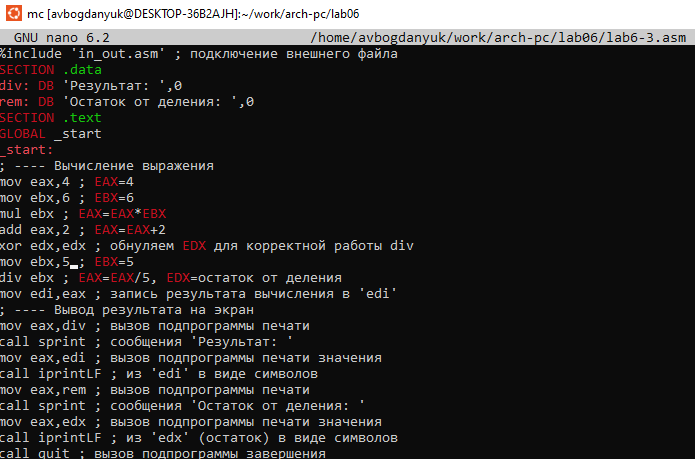


Figure 15: Файл lab6-3.asm

Создаю исполнительный файл и запускаю его (рис. [16](#fig:016)).

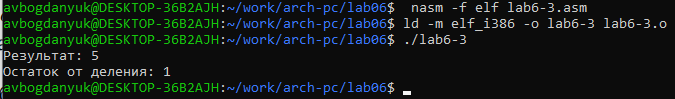


Figure 16: Создание и вывод

Создаю файл variant.asm и ввожу в него текст программы из листинга (рис. [17](#fig:017)).

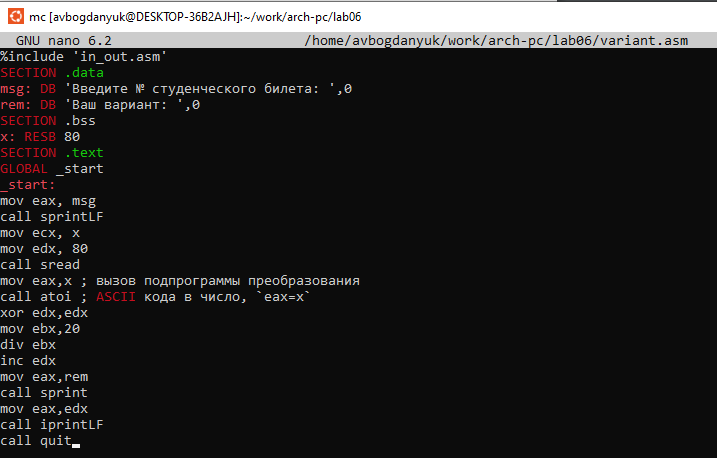


Figure 17: Файл variant.asm

Листинг:

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите № студенческого билета: ',0  
rem: DB 'Ваш вариант: ',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`  
xor edx,edx  
mov ebx,20  
div ebx  
inc edx  
mov eax,rem  
call sprint  
mov eax,edx  
call iprintLF  
call quit

Создаю исполнительный файл и запускаю его. Программа определила номер варианта 4 (рис. [18](#fig:018)).

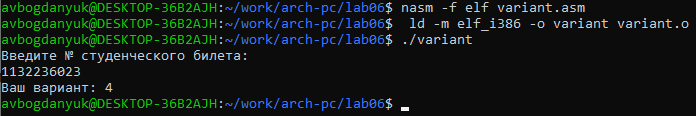


Figure 18: Создание и вывод

1)За вывод сообщения “Ваш вариант” отвечают строки: ’‘’NASM mov eax,rem call sprint’’’

2)mov ecx,x используется, чтобы положить адрес вводимой строки x в ecx. mov edx,80 используется для записи в edx длины вводимой строки. call sread для вызова подпрограммы из внешного файла для ввода текста с клавиатуры.

3)call atoi используется для вызова подпрограммы из внешного файла, что преобразает ascii-код символа в целое число и записывает результат в eax.

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

5)Остаток от деления при выполнении иструкции div ebx записывается в edx.

6)inc edx используется для увеличения значения edx на 1.

mov eax,edx  
call iprintLF

2.Задания для самостоятельной работы

Создаю файл lab6-4.asm и ввожу в файл текст программы для счета выражения (4/3)\*(x-1)+5(рис. [19](#fig:019)).

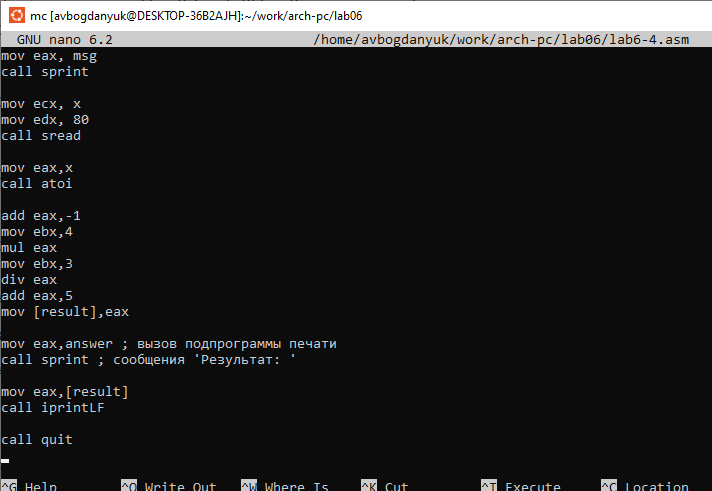


Figure 19: Файл lab6-4.asm

Листинг:

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите значение переменной x: ',0  
answer: DB 'Результат ',0  
  
SECTION .bss  
x: RESB 80  
result: RESB 80  
  
SECTION .text  
GLOBAL \_start  
  
\_start:  
mov eax, msg  
call sprint  
  
mov ecx, x  
mov edx, 80  
call sread  
  
mov eax,x  
call atoi  
  
add eax,-1  
mov ebx,4  
mul ebx  
mov ebx,3  
div ebx  
add eax,5  
mov [result],eax  
  
mov eax,answer ; вызов подпрограммы печати  
call sprint ; сообщения 'Результат: '  
  
mov eax,[result]  
call iprintLF  
  
call quit

Создаю исполнительный файл и запускаю его. Проверяю, ответы верные (рис. [20](#fig:020)).

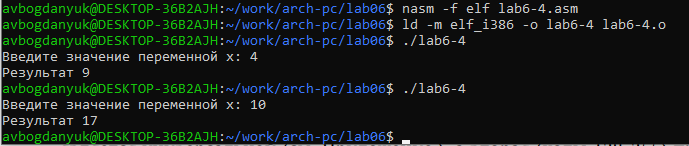


Figure 20: Создание и вывод

# Выводы

В ходе выполнения лабораторной работы я своила арифметические инструкции языка ассемблера NASM.

# Список литературы