Лабораторная работа №6

НКАбд-02-23

Выборнов Дмитрий Валерьевич

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. Вопросы по программе.
4. Задание для самостоятельной работы.

3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Три основных способа адресации: регистровая адресация, непосревственная адресация и адресация памяти. В **NASM** доступны команды сложения, вычетания, умножения, деления, изменения знака, инкремента и декремента. 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.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM.

Перехожу в каталог курса, создаю отдельный каталог для шестой лабораторной работы, перехожу в него и в нём создаю файл для первого задания.

```
dvvybornov@vvv-zenbook:~/work/arch-pc$ mkdir lab06
dvvybornov@vvv-zenbook:~/work/arch-pc$ cd lab06
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ touch lab6-1.asm
```

Рис. 4.1: Первый шаг.

Ввожу текст программы вывода регистра eax.

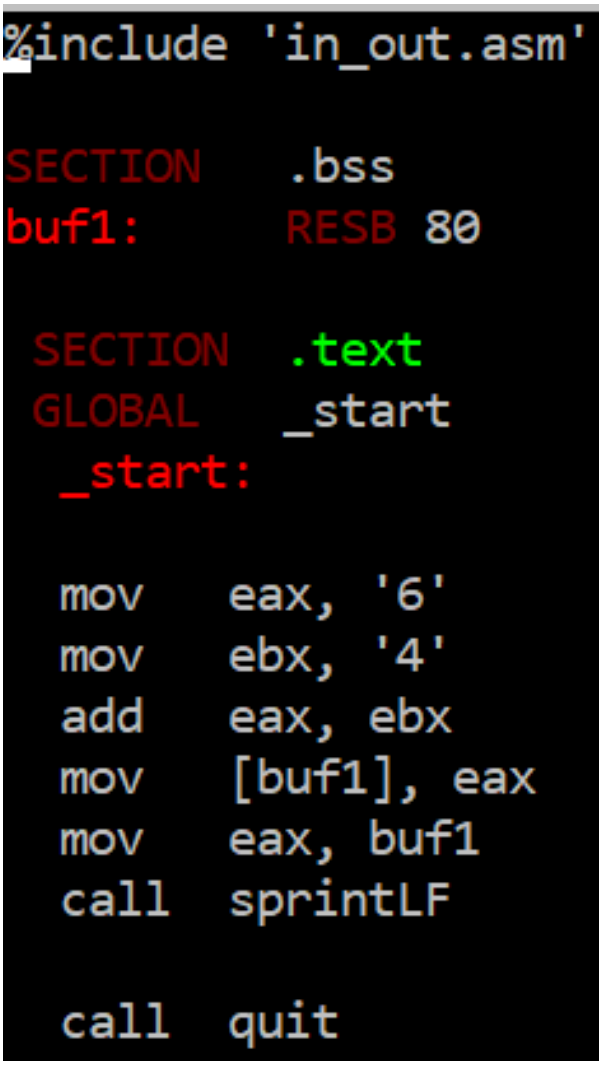

Рис. 4.2: Второй шаг.

Создаю исполняемый файл и запускаю его.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ./lab6-1
j
```

Рис. 4.3: Третий шаг.

Изменяю текст программы.

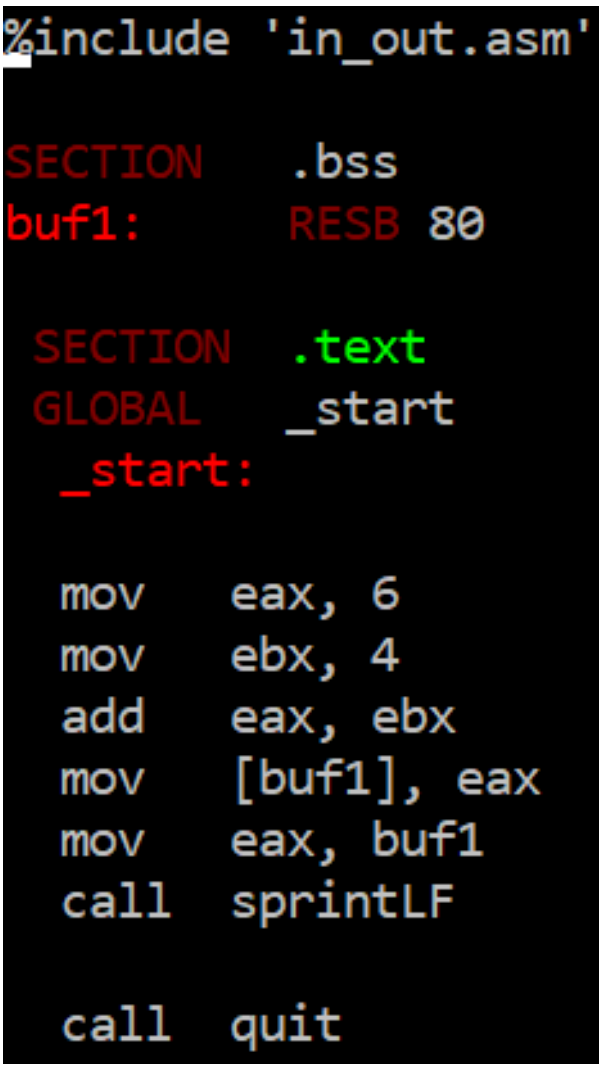

Рис. 4.4: Четвёртый шаг.

Запускаю изменённую программу.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ./lab6-1
```

Рис. 4.5: Пятый шаг.

На экране ничего не отображается, так как 10 - это код символа перевода строки.

Создаю новый файл и ввожу в него другую версию программы вывода регистра eax.

```
%include 'in_out.asm'

SECTION   .text
GLOBAL    _start
 _start:

 mov   eax, '6'
 mov   ebx, '4'
 add   eax, ebx
 call  iprintLF

 call quit
```

Рис. 4.6: Шестой шаг.

Создаю исполняемый файл новой версии программы и запускаю его.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ./lab6-2
106
```

Рис. 4.7: Седьмой шаг.

Изменяю новую версию программы аналогично предыдущей.

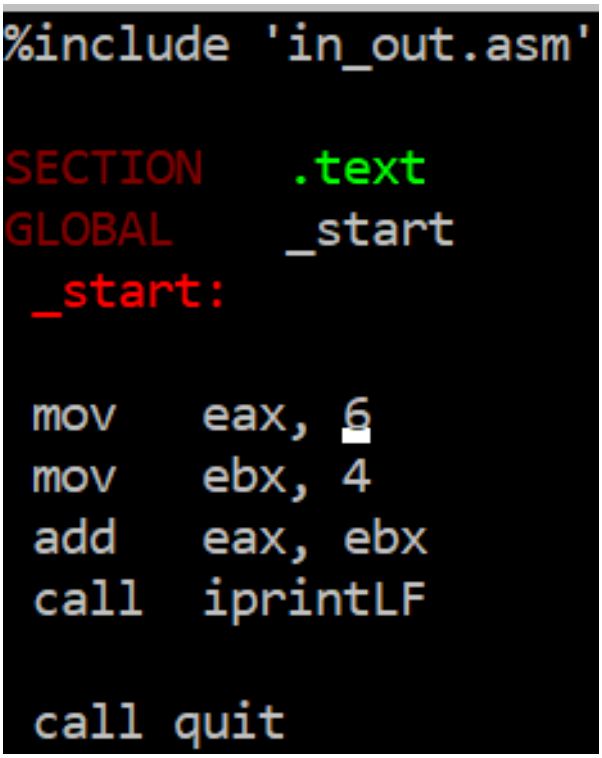

Рис. 4.8: Восьмой шаг.

Запускаю изменённую программу.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ./lab6-2
10
```

Рис. 4.9: Девятый шаг.

Программа выводит 10, т. к. теперь она складывает сами числа, а не их коды.

Заменяю функцию iprintLF на iprint.

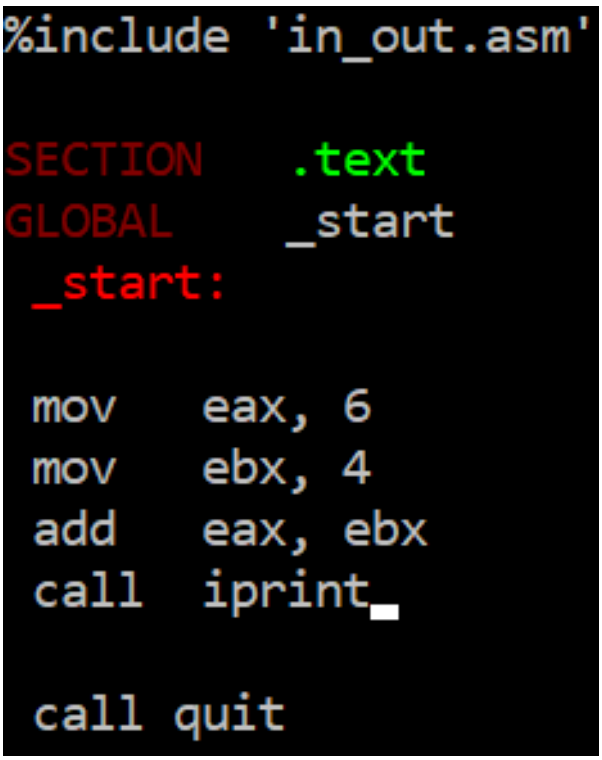

Рис. 4.10: Десятый шаг.

Запускаю полученную программу.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ./lab6-2
10dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$
```

Рис. 4.11: Одиннадцатый шаг.

Вывод функции iprintLF отличается от вывода функции iprint тем, что она добавляет к выводу символ переноса строки.

4.2 Выполнение арифметических операций в NASM.

Создаю новый файл lab6-3.asm и ввожу в него текст программы вычисления выражения.

```
%include 'in_out.asm'

SECTION   .data

div:   DB 'Результат: ' , 0
rem:   DB 'Остаток от деления: ' , 0

SECTION   .text
GLOBAL    _start
 _start:

 mov   eax, 5
 mov   ebx, 2
 mul   ebx
 add   eax, 3
 xor   edx, edx
 mov   ebx, 3
 div   ebx

 mov   edi, eax

 mov   eax, div
 call  sprint
 mov   eax, edi
 call  iprintLF

 mov   eax, rem
 call  sprint
 mov   eax, edx
 call  iprintLF

 call  quit
```

Рис. 4.12: Двенадцатый шаг.

Создаю исполняемый файл и запускаю его.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.13: Тринадцатый шаг.

Изменяю текст программы для вычисления выражения F(x) = (4 * 6 + 2)/5.

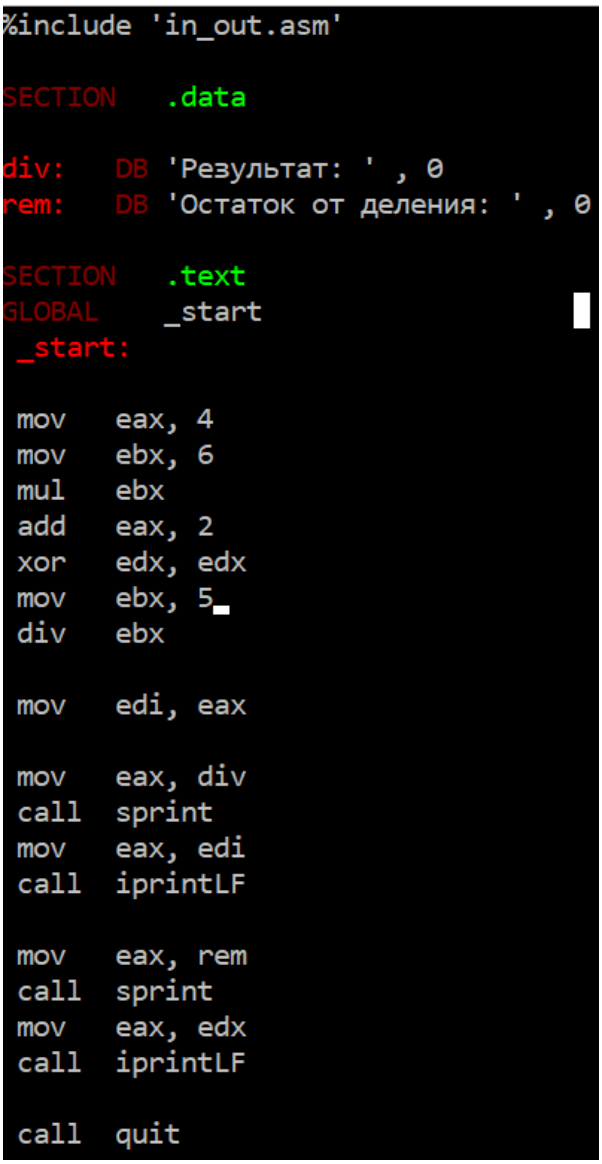

Рис. 4.14: Четырнадцатый шаг.

Запускаю изменённую программу.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рис. 4.15: Пятнадцатый шаг.

Создаю новый файл и ввожу в него программу для вычисления варианта задания по номеру студенческого билета.

```
%include 'in_out.asm'

SECTION .data
msg:   DB 'Введите № студеньческого 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
 call  atoi
 xor   edx, edx
 mov   ebx, 20
 div   ebx
 inc   edx

 mov   eax, rem
 call  sprint
 mov   eax, edx
 call  iprintLF

 call  quit
```

Рис. 4.16: Шестнадцатый шаг.

Создаю исполняемый файл и запускаю его.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ nasm -f elf variant.asm
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ./variant
Введите № студеньческого билета:
1132236029
Ваш вариант: 10
```

Рис. 4.17: Семнадцатый шаг.

4.3 Ответы на вопросы по программе.

1. За вывод на экран сообщения “Ваш вариант:” отвечают строки **mov eax,** **rem** и **call sprint**.
2. Инструкция **mov eax, x** используется для определения адреса записи значения вводимой строки x в регистр eax. Инструкция **mov edx, 80** используется для записи длины вводимой строки в регистр edx, а инструкция **call sread** - для вызова подпрограммы из внешнего файла, которая используется для ввода с клавиатуры.
3. Строка **call atoi** используется для вызова подпрограммы из внешнего файла, которая преобразует ascii - код символа в целое число.
4. За вычисления варианта отвечают строки **xor edx, edx**, **mov edx, 20**, **div ebx** и **inc edx**.
5. Остаток записывается в регистр **edx**.
6. Инструкция **inc edx** увеличивает значение регистра **edx** на 1.
7. За вывод результата вычислений отвечают строки **mov eax, edc** и **call iprintLF**.

4.4 Задание для самостоятельной работы.

Создаю файл Expression.asm и ввожу в него текст программы для выражения №10.

```
%include 'in_out.asm'

SECTION .data
msg:    DB '5(x + 18) - 28' , 0
rem:    DB 'Введите значение x: ' , 0

SECTION .bss
x:    RESB 80

SECTION .text
GLOBAL _start
 _start:

 mov   eax, msg
 call  sprintLF

 mov   eax, rem
 call  sprintLF

 mov   ecx, x
 mov   edx, 80
 call  sread

 mov   eax, x
 call  atoi
 xor   edx, edx
 add   eax, 18
 mov   edx, 5
 mul   edx
 sub   eax, 28

 mov   edi, eax

 mov   eax, edi
 call  iprintLF

 call  quit
```

Рис. 4.18: Первый шаг задания для самостоятельной работы.

Создаю исполняемый файл, запускаю его и проверяю его работу для значений x = 2 и x = 3.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ nasm -f elf Expression.asm
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o Expression Expression.o
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ./Expression
5(x + 18) - 28
Введите значение x:
2
72
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$ ./Expression
5(x + 18) - 28
Введите значение x:
3
77
dvvybornov@vvv-zenbook:~/work/arch-pc/lab06$
```

Рис. 4.19: Второй шаг задания для самостоятельной работы.

5 Выводы

Выполнив эту лабораторную работу, я освоил арифметические инструкции языка ассемблера **NASM**.