

Отчёт по лабораторной работе № 6

НБИбд-01-22

Иовков Мирослав Алексеевич

Содержание

1	Цель работы	4
2	Задание	5
3	Порядок выполнения лабораторной работы	6
3.1	Символьные и численные данные в NASM	6
3.2	Выполнение арифметических операций в NASM	10
3.3	Задание для самостоятельной работы	15
4	Выводы	16

Список иллюстраций

3.1	Файл lab7-1.asm	6
3.2	Текст программы из листинга 7.1	7
3.3	Исполняемый файл	7
3.4	Текст программы	7
3.5	Исполняемый файл	8
3.6	файл lab7-2.asm	8
3.7	Текст программы из листинга 7.2	8
3.8	Исполняемый файл	9
3.9	Изменённый код	9
3.10	Результат программы	9
3.11	iprintLF на iprint	10
3.12	Работа исполняемого файла	10
3.13	Файл lab7-3.asm	10
3.14	Текст программы из листинга 7.3	11
3.15	Изменённый текст программы	11
3.16	Исполняемый файл	12
3.17	Файл variant.asm	13
3.18	Текст программы из листинга 7.4	13
3.19	Работа исполняемого файла	14
3.20	Файл variant.asm	15

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. выполнить работу с символьными и численными данными в NASM
2. Отработать на практике арифметические операции в NASM
3. Написать программу вычисления выражения с входными данными

3 Порядок выполнения лабораторной работы

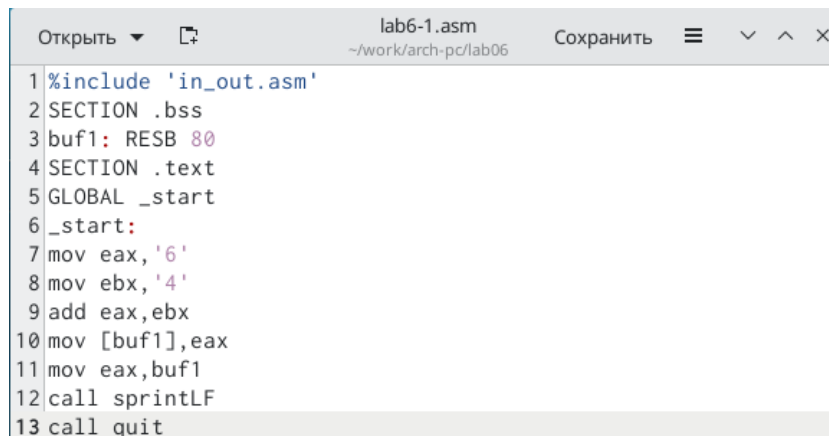
3.1 Символьные и численные данные в NASM

1. Создали каталог для программ лабораторной работы № 7, перешли в него и создайте файл lab7-1.asm. (рис. 3.1)

```
maiovkov@dk3n40 ~ $ mkdir ~/work/arch-pc/lab06
maiovkov@dk3n40 ~ $ cd ~/work/arch-pc/lab06
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ touch lab6-1.asm
maiovkov@dk3n40 ~/work/arch-pc/lab06 $
```

Рис. 3.1: Файл lab7-1.asm

2. Рассмотрели примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр еах. Ввели в файл lab7-1.asm текст программы из листинга 7.1. (рис. 3.2) В данной программе в регистр еах записывается символ 6 (mov еах,'6'), в регистр еbх символ 4 (mov еbх,'4'). Далее к значению в регистре еах прибавляется значение регистра еbх (add еах,еbх, результат сложения запишется в регистр еах). Далее выводится результат. Так как для работы функции sprintLF в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого записали значение регистра еах в переменную buf1 (mov [buf1],еах), а затем записали адрес переменной buf1 в регистр еах (mov еах,buf1) и вызвали функцию sprintLF.



```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call printf
13 call quit
```

Рис. 3.2: Текст программы из листинга 7.1

Создали исполняемый файл и запустили его. (рис. 3.3)

```
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ./lab6-1
j
```

Рис. 3.3: Исполняемый файл

3. Далее изменили текст программы и вместо символов, записали в регистры числа. (рис. 3.4)



```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, 6
6 mov ebx, 4
7 add eax, ebx
8 call iprintf
9 call quit
```

Рис. 3.4: Текст программы

Создали исполняемый файл и запустили его. (рис. 3.5)

```

maiovkov@dk3n40 ~/work/arch-pc/lab06 $ touch lab6-1.asm
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ./lab6-1

maiovkov@dk3n40 ~/work/arch-pc/lab06 $

```

Рис. 3.5: Исполняемый файл

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определили, что код 10 соответствует символ /n. Это символ перевода строки, он не отображается.

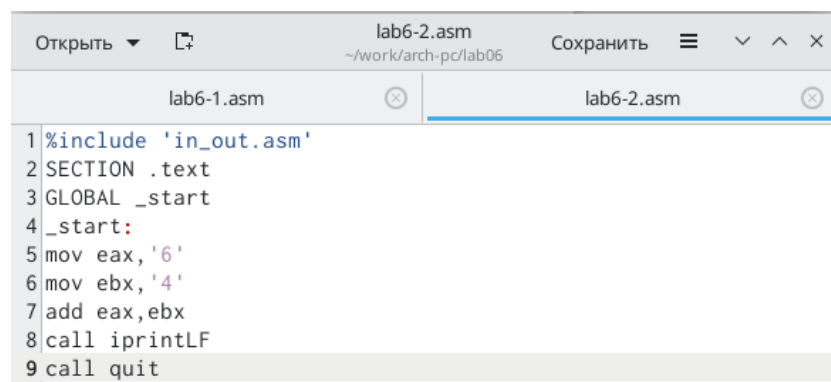
4. Как отмечалось выше, для работы с числами в файле in_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовали текст программы из Листинга 7.1 с использованием этих функций. Создали файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 (рис. 3.6) и ввели в него текст программы из листинга 7.2. (рис. 3.7)

```

maiovkov@dk3n40 ~/work/arch-pc/lab06 $ touch lab6-2.asm

```

Рис. 3.6: файл lab7-2.asm



```

lab6-2.asm
~/work/arch-pc/lab06
Сохранить
lab6-1.asm
lab6-2.asm
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, '6'
6 mov ebx, '4'
7 add eax, ebx
8 call iprintLF
9 call quit

```

Рис. 3.7: Текст программы из листинга 7.2

Создали исполняемый файл и запустили его. (рис. 3.8)


```

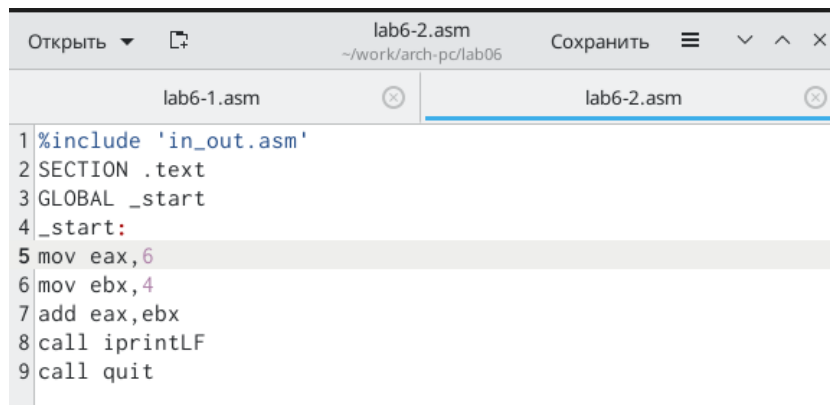
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ touch lab6-2.asm
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ./lab6-2
106

```

Рис. 3.8: Исполняемый файл

В результате работы программы мы получили число 106. В данном случае, как и в первом, команда `add` складывает коды символов '6' и '4' ($54+52=106$). Однако, в отличие от программы из листинга 7.1, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

5. Аналогично предыдущему примеру изменим символы на числа. (рис. 3.9)



```

lab6-2.asm
~/work/arch-pc/lab06
Сохранить
lab6-1.asm lab6-2.asm
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprintLF
9 call quit

```

Рис. 3.9: Изменённый код

Создайте исполняемый файл и запустите его. В результате при исполнении программы получили 10. (рис. 3.10)

```

maiovkov@dk3n40 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ./lab6-2
10

```

Рис. 3.10: Результат программы

Заменили функцию `iprintLF` на `iprint`. (рис. 3.11) Создайте исполняемый файл и запустите его. (рис. 3.12) Вывод функций `iprintLF` и `iprint` отличается наличием перевода строки после вывода?



```
8 call iprint
```

Рис. 3.11: iprintLF на iprint

```
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 3.12: Работа исполняемого файла

3.2 Выполнение арифметических операций в NASM

6. В качестве примера выполнения арифметических операций в NASM привели программу вычисления арифметического выражения $\boxtimes(\boxtimes) = (5 \boxtimes 2 + 3)/3$. Создали файл lab7-3.asm в каталоге ~/work/arch-pc/lab07. (рис. 3.13)



```
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ touch lab6-3.asm
```

Рис. 3.13: Файл lab7-3.asm

Внимательно изучили текст программы из листинга 7.3 и ввели в lab7-3.asm. (рис. 3.14)

```

1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4 %include 'in_out.asm' ; подключение внешнего файла
5 SECTION .data
6 div: DB 'Результат: ',0
7 rem: DB 'Остаток от деления: ',0
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ; ---- Вычисление выражения
12 mov eax,5 ; EAX=5
13 mov ebx,2 ; EBX=2
14 mul ebx ; EAX=EAX*EBX
15 add eax,3 ; EAX=EAX+3
16 xor edx,edx ; обнуляем EDX для корректной работы div
17 mov ebx,3 ; EBX=3
18 div ebx ; EAX=EAX/3, EDX=остаток от деления
19 mov edi,eax ; запись результата вычисления в 'edi'
20 ; ---- Вывод результата на экран
21 mov eax,div ; вызов подпрограммы печати
22 call sprint ; сообщения 'Результат: '
23 mov eax,edi ; вызов подпрограммы печати значения
24 call iprintLF ; из 'edi' в виде символов
25 mov eax,rem ; вызов подпрограммы печати
26 call sprint ; сообщения 'Остаток от деления: '
27 mov eax,edx ; вызов подпрограммы печати значения
28 call iprintLF ; из 'edx' (остаток) в виде символов
29 call quit ; вызов подпрограммы завершения

```

Рис. 3.14: Текст программы из листинга 7.3

Создали исполняемый файл и запустили его. Получили следующий результат.
(рис. ??)

Исполняемый файл

Изменили текст программы для вычисления выражения $(4 \times 6 + 2)/5$.
(рис. 3.15)

```

maiovkov@dk3n40 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1

```

Рис. 3.15: Изменённый текст программы

Создали исполняемый файл и проверили его работу. (рис. 3.16)

```
maiovkov@dk3n40 ~/work/arch-pc/lab06 $ touch variant.asm
```

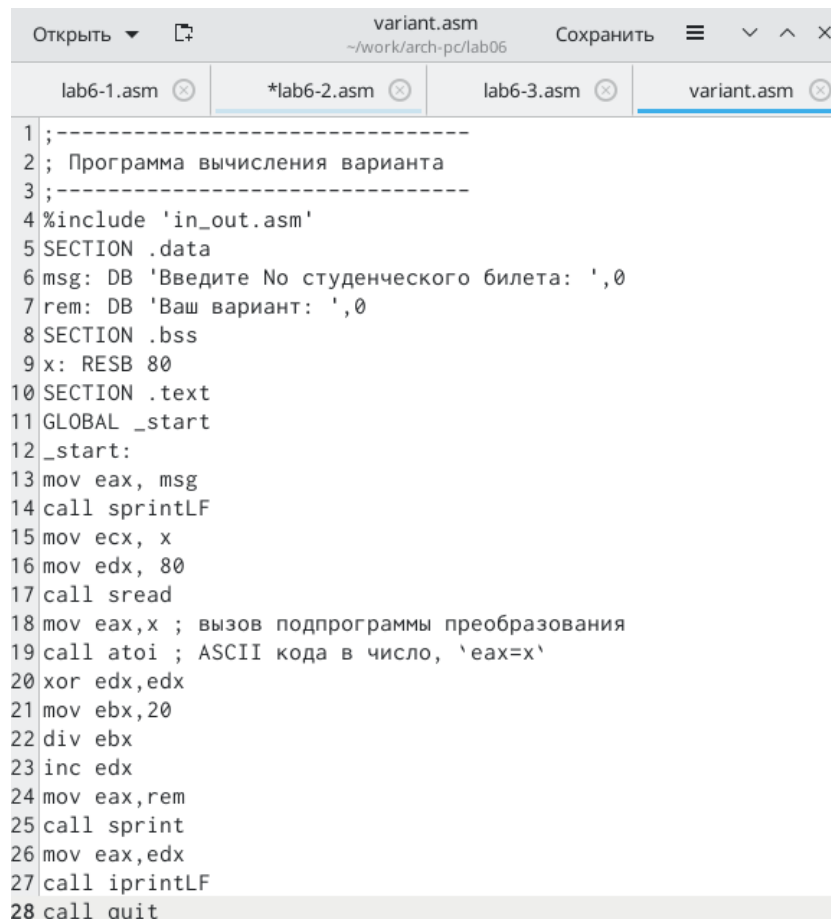
Рис. 3.16: Исполняемый файл

7. В качестве другого примера рассмотрели программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

- вывести запрос на введение № студенческого билета
- вычислить номер варианта по формуле: $(\text{№} \bmod 20) + 1$, где № – номер студенческого билета (В данном случае $\text{№} \bmod \text{№}$ – это остаток от деления № на №).
- вывести на экран номер варианта.

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше, ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы преобразуются в числа. Для этого использована функция `atoi` из файла `in_out.asm`.

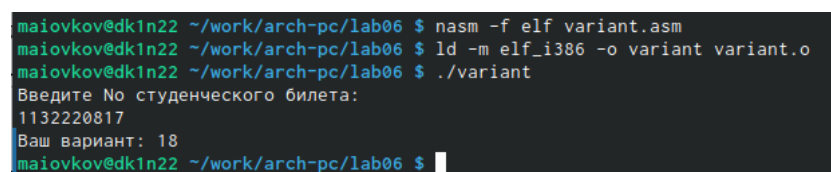
Создали файл `variant.asm` в каталоге `~/work/arch-pc/lab07`. (рис. 3.17)



```
1 ;-----
2 ; Программа вычисления варианта
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg: DB 'Введите No студенческого билета: ',0
7 rem: DB 'Ваш вариант: ',0
8 SECTION .bss
9 x: RESB 80
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprintf
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax, x ; вызов подпрограммы преобразования
19 call atoi ; ASCII кода в число, 'eax=x'
20 xor edx, edx
21 mov ebx, 20
22 div ebx
23 inc edx
24 mov eax, rem
25 call sprintf
26 mov eax, edx
27 call iprintf
28 call quit
```

Рис. 3.17: Файл variant.asm

Внимательно изучили текст программы из листинга 7.4 и ввели в файл variant.asm. (рис. 3.18)



```
maiovkov@dk1n22 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
maiovkov@dk1n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
maiovkov@dk1n22 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132220817
Ваш вариант: 18
maiovkov@dk1n22 ~/work/arch-pc/lab06 $
```

Рис. 3.18: Текст программы из листинга 7.4

Создали исполняемый файл и запустили его. (рис. 3.19) Проверили результат работы программы вычислив номер варианта аналитически.

```

5 rem2: DB '3*(x+10)-20',0
6 SECTION .bss
7 x: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 mov eax, rem2
13 call sprintLF
14 mov eax, rem1
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax,x ; вызов подпрограммы преобразования
20 call atoi
21
22 mov eax,x
23 call atoi
24 add eax,10
25 xor edx,edx
26 mov ebx,3
27 mul ebx
28 xor edx,edx
29 mov ebx,20
30 neg ebx
31 add eax, ebx
32 mov edi,eax
33 ; -- Вывод результата на экран
34 mov eax,div ; вызов подпрограммы печати
35 call sprint ; сообщения 'Результат: '
36 mov eax,edi ; вызов подпрограммы печати значения
37 call iprintLF ; из 'edi' в виде символов
38 call quit ; вызов подпрограммы завершения

```

Рис. 3.19: Работа исполняемого файла

Ответы на вопросы лабораторной работы: 1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения 'Ваш вариант: '? `mov eax,rem call sprint` 2. Для чего используются следующие инструкции? `mov ecx, x` - запись входной переменной в регистр `ecx`; `mov edx, 80` - запись размера переменной в регистр `edx`; `call sread` - вызов процедуры чтения данных; 3. Для чего используется инструкция "call atoi"? Вызов `atoi` – функции преобразующей `ascii`-код символа в целое число и записывающий результат в регистр `eax`. 4. Какие строки листинга 7.4 отвечают за вычисления варианта? `xor edx,edx mov ebx,20 div ebx inc edx` 5. В какой регистр записывается остаток от деления при выполнении инструкции "div ebx"? В регистр `ebx`. 6. Для чего используется инструкция "inc edx"? Увеличивает значение `edx` на 1. 7. Какие строки листинга 7.4 отвечают за вывод на экран результата

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

3.3 Задание для самостоятельной работы

Написали программу вычисления выражения $y = x(x)$. Программа выводит выражение для вычисления, выводит запрос на ввод значения x , вычисляет заданное выражение в зависимости от введенного x , выводит результат вычислений. Вид функции $x(x)$ выбрали из таблицы 6.3 вариантов заданий, наш номер - 18, полученный при выполнении лабораторной работы. Создали исполняемый файл и проверили его работу для значений $x_1 = 1$ и $x_2 = 5$ из 6.3.

```
maiovkov@dk1n22 ~/work/arch-pc/lab06 $ nasm -f elf variant18.asm
maiovkov@dk1n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant18 variant18.o
maiovkov@dk1n22 ~/work/arch-pc/lab06 $ ./variant18
3*(x+10)-20
Введите x: 1
Результат: 13
maiovkov@dk1n22 ~/work/arch-pc/lab06 $ ./variant18
3*(x+10)-20
Введите x: 5
Результат: 25
```

Рис. 3.20: Файл variant.asm

Работа исполняемого файла

4 Выводы

В результате выполнения лабораторной работы были освоены арифметические инструкции языка ассемблера NASM.