

# **Отчёт по лабораторной работе №7**

**Дисциплина: Архитектура компьютера**

Гибшер Кирилл Владимирович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>16</b>
	<b>Список литературы</b>	<b>17</b>

## Список иллюстраций

4.1	Создание каталога lab7-1.asm и файла для работы . . . . .	8
4.2	Текст программы . . . . .	9
4.3	Создание исполняемого файла и запуск программы . . . . .	9
4.4	Изменённый текст программы . . . . .	10
4.5	Вывод изменённой программы . . . . .	10
4.6	Программа для вывода значения регистра eax . . . . .	10
4.7	Вывод программы(файл lab7-2) . . . . .	11
4.8	Замена в тексте программы “6” и “4” на числа 6 и 4 . . . . .	11
4.9	Запуск изменённой программы в файле lab7-2 . . . . .	11
4.10	Замена функции iprintLF на iprint . . . . .	12
4.11	Программа по вычислению выражения $f(x) = (5 * 2 + 3)/3$ . . . . .	12
4.12	Программа по вычислению выражения $f(x) = (4 * 6 + 2)/5$ . . . . .	12
4.13	Создание исполняемого файла и вывод программы . . . . .	13
4.14	Текст программы по вычислению варианта . . . . .	13
4.15	Вывод программы из файла variant.asm . . . . .	14
4.16	Текст программы для вычисления $(8x-6)/2$ . . . . .	15
4.17	Вывод программы по вычислению $(8x-6)/2$ . . . . .	15

## **Список таблиц**

# 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. 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.

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`.

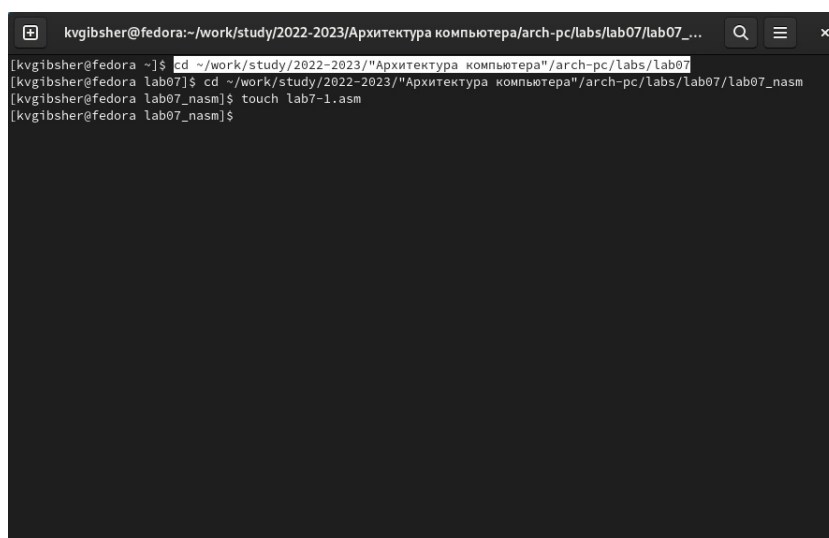
Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

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.

## 4 Выполнение лабораторной работы

1. Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm: (рис. 4.1)



```
kvg1bsher@fedora:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab07/lab07_...  
[kvg1bsher@fedora ~]$ cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab07  
[kvg1bsher@fedora lab07]$ cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab07/lab07_nasm  
[kvg1bsher@fedora lab07_nasm]$ touch lab7-1.asm  
[kvg1bsher@fedora lab07_nasm]$
```

Рис. 4.1: Создание каталога lab7-1.asm и файла для работы

2. Открываю файл lab7-1.asm и вставляю в него программу вывода значений, записанных в регистр eax (рис. 4.2)



```

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 sprintf
13 call quit
14
15

```

Рис. 4.2: Текст программы

3. Создаю исполняемый файл и запускаю его, предварительно добавив в каталог подключаемый файл `in_out.asm`. Результатом будет символ `j`. Это происходит потому, что код символа `6` равен `00110110` в двоичном представлении, а код символа `4` – `00110100`. Команда `add eax, ebx` запишет в регистр `eax` сумму кодов – `01101010` (`106` в десятичном представлении), что в свою очередь является кодом символа `j` по таблице ASCII (рис. 4.3)

```

[kvgibsher@fedora lab07_nasm]$ nasm -f elf lab7-1.asm
[kvgibsher@fedora lab07_nasm]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[kvgibsher@fedora lab07_nasm]$ ./lab7-1
j
[kvgibsher@fedora lab07_nasm]$

```

Рис. 4.3: Создание исполняемого файла и запуск программы

4. Заменяю в тексте программы символы “6” и “4” на цифры 6 и 4(рис. 4.4)

```

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 sprintLF
13 call quit
14
15

```

Рис. 4.4: Изменённый текст программы

5. Создаю новый исполняемый файл программы и запускаю его. Результатом является пустой вывод(так как вывелся символ с кодом 10, а это символ перевода строки, этот символ не отображается при выводе на экран (рис. 4.5))

```

[kvg1bshergfedora lab07_nasm]$ nasm -f elf lab7-1.asm
[kvg1bshergfedora lab07_nasm]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[kvg1bshergfedora lab07_nasm]$ ./lab7-1
[kvg1bshergfedora lab07_nasm]$

```

Рис. 4.5: Вывод изменённой программы

6. Создаю файл lab7-2.asm с помощью утилиты touch. Ввожу в файл текст программы для вывода значения регистра eax (рис. 4.6)

```

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
10

```

Рис. 4.6: Программа для вывода значения регистра eax

7. Создаю исполняемый файл и запускаю его. Результатом является число 106(это происходит потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”) (рис. 4.7)

```
[kvgibsher@fedora lab07_nasm]$ nasm -f elf lab7-2.asm
[kvgibsher@fedora lab07_nasm]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[kvgibsher@fedora lab07_nasm]$ ./lab7-2
106
[kvgibsher@fedora lab07_nasm]$
```

Рис. 4.7: Вывод программы(файл lab7-2)

8. Заменяю в тексте программы в файле lab7-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.8)

```
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
10
```

Рис. 4.8: Замена в тексте программы “6” и “4” на числа 6 и 4

9. Создаю исполняемый файл и запускаю его. Результатом является число 106(это происходит потому что теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10)(рис. 4.9)

```
[kvgibsher@fedora lab07_nasm]$ nasm -f elf lab7-2.asm
[kvgibsher@fedora lab07_nasm]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[kvgibsher@fedora lab07_nasm]$ ./lab7-2
10
[kvgibsher@fedora lab07_nasm]$
```

Рис. 4.9: Запуск изменённой программы в файле lab7-2

10. Заменяю в тексте программы функцию `iprintLF` на `iprint`. Создаю исполняемый файл и запускаю его. Результатом является исчезновение переноса строки (рис. 4.10)

```
[kvgibsher@fedora lab07_nasm]$ nasm -f elf lab7-2.asm
[kvgibsher@fedora lab07_nasm]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[kvgibsher@fedora lab07_nasm]$ ./lab7-2
10[kvgibsher@fedora lab07_nasm]$
```

Рис. 4.10: Замена функции `iprintLF` на `iprint`

11. Создаю файл `lab7-3.asm` с помощью утилиты `touch`. Ввожу в созданный файл текст программы для вычисления значения выражения  $f(x) = (5 * 2 + 3)/3$ . Создаю исполняемый файл и запускаю его. Получаю корректный результат (рис. 4.11)

```
[kvgibsher@fedora lab07_nasm]$ nasm -f elf lab7-3.asm
[kvgibsher@fedora lab07_nasm]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[kvgibsher@fedora lab07_nasm]$ ./lab7-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.11: Программа по вычислению выражения  $f(x) = (5 * 2 + 3)/3$

12. Изменяю программу так, чтобы она вычисляла значение выражения  $f(x) = (4 * 6 + 2)/5$  (рис. 4.12)

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

Рис. 4.12: Программа по вычислению выражения  $f(x) = (4 * 6 + 2)/5$

13. Создаю исполняемый файл и запускаю его. Вывод результата и остатка от деления. Программа отработала верно (рис. 4.13)

```
[kvgibsher@fedora lab07_nasm]$ nasm -f elf lab7-3.asm
[kvgibsher@fedora lab07_nasm]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[kvgibsher@fedora lab07_nasm]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[kvgibsher@fedora lab07_nasm]$
```

Рис. 4.13: Создание исполняемого файла и вывод программы

14. Создаю файл variant.asm с помощью утилиты touch. Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.14)

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

Рис. 4.14: Текст программы по вычислению варианта

Ответы на вопросы:

- 1) За вывод на экран сообщения 'Ваш вариант:' отвечают строки `mov eax, rem`  
`call sprint`
- 2) `mov ecx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx` `mov edx, 80` - запись в регистр `edx` длины вводимой строки `call`

sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры

- 3) Инструкция `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 вычисление варианта отвечают строки `xor edx,edx mov ebx,20 div ebx inc edx`
  - 5) Остаток от деления при выполнении инструкции `"div ebx"` записывается в регистр `edx`
  - 6) Инструкция `"inc edx"` используется для увеличения значения регистра `edx` на 1
  - 7) За вывод на экран результата вычислений отвечают строки `mov eax,edx call iprintLF`
15. Создаю исполняемый файл и запускаю его. Получаю 12 номер варианта. (рис. 4.15)

```
[kvgibsher@fedora lab07_nasm]$ nasm -f elf variant.asm
[kvgibsher@fedora lab07_nasm]$ ld -m elf_i386 -o variant variant.o
[kvgibsher@fedora lab07_nasm]$ ./variant
Введите № студенческого билета:
1132221811
Ваш вариант: 12
[kvgibsher@fedora lab07_nasm]$
```

Рис. 4.15: Вывод программы из файла `variant.asm`

16. Приступаю к выполнению заданий для самостоятельной работы. Создаю файл `lab7-4.asm` с помощью утилиты `touch`. Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения под номером 12:  $(8x-6)/2$  (рис. 4.16)

```

1:-----
2: Программа вычисления функции
3:-----
4%include 'in_out.asm'
5
6 SECTION .data
7 msg: DB 'Введите значение x: ',0
8 rem: DB 'Ваш результат: ',0
9 SECTION .bss
10 x: RESB 80
11
12 SECTION .text
13 GLOBAL _start
14 _start:
15
16 mov eax, msg
17 call sprintf
18
19 mov ecx, x
20 mov edx, 80
21 call sread
22
23 mov eax, x ; вызов подпрограммы преобразования
24 call atoi ; ASCII кода в число, 'eax-x'
25
26 mov ebx, 8
27 mul ebx
28
29 mov ebx, 6
30 neg ebx
31 add eax, ebx
32
33 xor edx, edx
34 xor ebx, ebx
35 mov ebx, 2
36 div ebx
37

```

Рис. 4.16: Текст программы для вычисления  $(8x-6)/2$

17. Создаю исполняемый файл и запускаю его. При вводе  $x = 1$  получаю ответ 1, при вводе  $x = 5$  получаю ответ 17. Делаю проверку вручную. Оба полученных значения являются корректными (рис. 4.17)

```

[kvgibsher@fedora lab07_nasm]$ ./lab7-4
Введите значение x:
1
Ваш результат: 1
[kvgibsher@fedora lab07_nasm]$ 5
bash: 5: команда не найдена...
[kvgibsher@fedora lab07_nasm]$ ./lab7-4
Введите значение x:
5
Ваш результат: 17

```

Рис. 4.17: Вывод программы по вычислению  $(8x-6)/2$

## 5 Выводы

При выполнении лабораторной работы освоил арифметические инструкции языка ассемблера NASM.



## **Список литературы**