

Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютеров и операционные системы

Авдадаев Джамал Геланиевич

Содержание

1	Цель работы	1
2	Задание	1
3	Теоретическое введение.....	1
4	Выполнение лабораторной работы.....	3
4.1	Символьные и численные данные в NASM	3
4.2	Выполнение арифметических операций в NASM	6
4.2.1	Ответы на вопросы по листингу 6.4.....	8
4.3	Задание для самостоятельной работы.....	9
5	Выводы.....	10
6	Список литературы.....	10

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. Ответы на вопросы по листингу 6.4
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.

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. Допустимые сочетания операндов для команды `add` аналогичны сочетаниям операндов для команды `mov`. Так, например, команда `add eax,ebx` прибавит значение из регистра `eax` к значению из регистра `ebx` и запишет результат в регистр `eax`. 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`.

Например, в табл. 1 приведено краткое описание стандартных каталогов Unix.

Table 1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [1–6].

4 Выполнение лабораторной работы

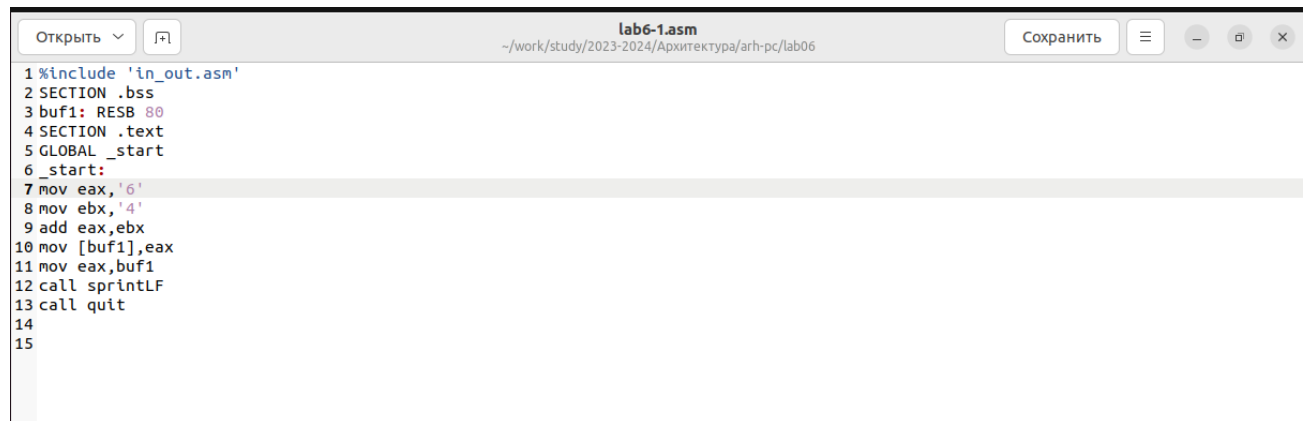
4.1 Символьные и численные данные в NASM

Создаю каталог для программ лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm. (рис. 20).

```
dgavdadaev@dgavdadaev-laptop:~$ mkdir ~/work/study/2023-2024/Архитектура/arh-pc/lab06
dgavdadaev@dgavdadaev-laptop:~$ cd ~/work/study/2023-2024/Архитектура/arh-pc/lab06
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ touch lab6-1.asm
```

Figure 1: Создание каталога и файла

Ввожу в файл lab6-1.asm текст программы из листинга 6.1. (рис. 20).



```
lab6-1.asm
~/work/study/2023-2024/Архитектура/arh-pc/lab06
Сохранить

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
14
15
```

Figure 2: Ввод текста программы

Создаю исполняемый файл и запускаю его. (рис. 20).

```
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ nasm -f elf lab6-1.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ ./lab6-1
6
j
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$
```

Figure 3: Запуск программы

Далее изменю текст программы и вместо символов запишу в регистры числа.

Исправляю текст программы следующим образом:

заменяю строки

mov eax, '6'

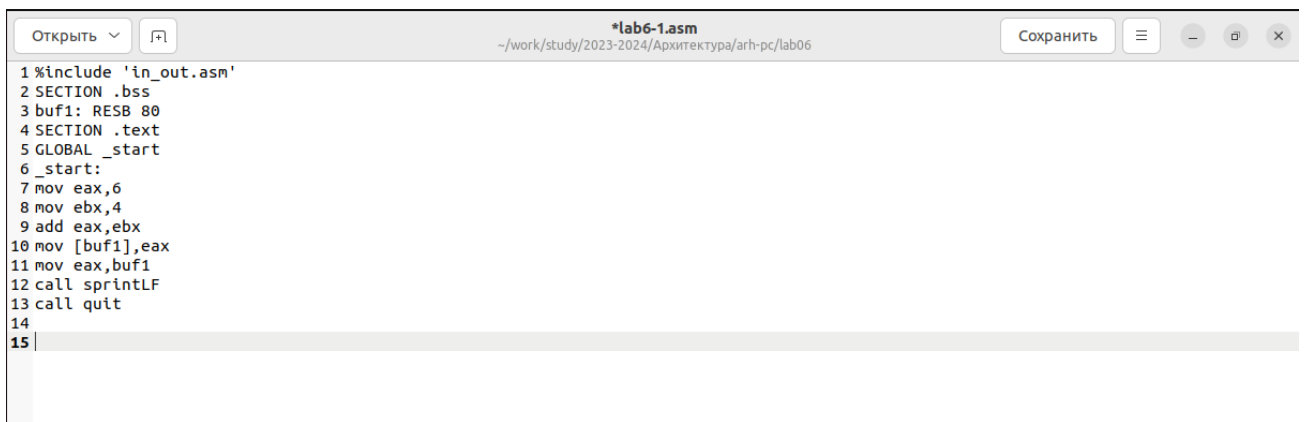
mov ebx, '4'

на строки

mov eax, 6

mov ebx, 4

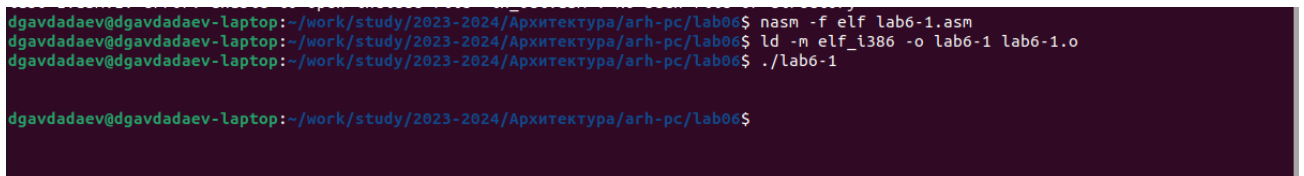
(рис. 20).



```
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
14
15
```

Figure 4: Замена некоторых строк кода

Создаю исполняемый файл и запускаю его. (рис. 20).



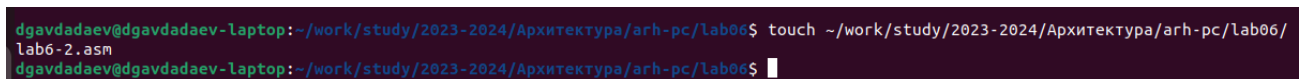
```
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ nasm -f elf lab6-1.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ ./lab6-1

dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$
```

Figure 5: Запуск кода

Данному коду (10) соответствует символ “LF, n”, который перемещает курсор на следующую строку. Сам символ при выводе на экран не отображается.

Создаю файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 (рис. 20).



```
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ touch ~/work/study/2023-2024/Архитектура/arch-pc/lab06/lab6-2.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$
```

Figure 6: Создание файла

и ввожу в него текст программы из листинга 6.2. (рис. 20).



```
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,'6'
8 mov ebx,'4'
9 add eax,ebx
10 call iprint
11
12 call quit
```

Figure 7: Ввод текста программы

Создаю исполняемый файл и запускаю его. (рис. 20).

```
lab6-2.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ nasm -f elf lab6-2.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ ./lab6-2
106
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$
```

Figure 8: Запуск исполняемого файла

В этой программе заменяю строки

`mov eax,'6'`


`mov ebx,'4'`

на строки

`mov eax,6`

`mov ebx,4`

(рис. 20).



```
*lab6-2.asm
~/work/study/2023-2024/Архитектура/arh-pc/lab06
Сохранить
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 call iprintLF
11
12 call quit
```

Figure 9: Изменение кода

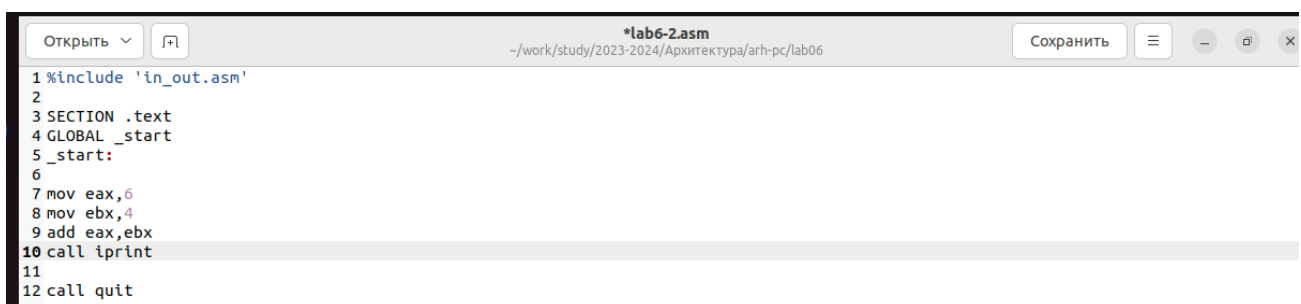
Создаю исполняемый файл и запускаю его. (рис. 20).

```
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ nasm -f elf lab6-2.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ ./lab6-2
10
```

Figure 10: Запуск исполняемого файла

В результате получаем число 10.

Заменяю функцию `iprintLF` на `iprint`. (рис. 20).



```
*lab6-2.asm
~/work/study/2023-2024/Архитектура/arh-pc/lab06
Сохранить
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 call iprint
11
12 call quit
```

Figure 11: Изменение кода

Создаю исполняемый файл и запускаю его. (рис. 20).

```
10
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ nasm -f elf lab6-2.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ ./lab6-2
10dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$
```

Figure 12: Запуск исполняемого файла

Вывод функций `iprintLF` и `iprint` отличается тем, что при использовании первой выполняется перенос на следующую строку после вывода, а при использовании второй этого не происходит.

4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06` и ввожу в него текст из листинга 6.3. (рис. 20).

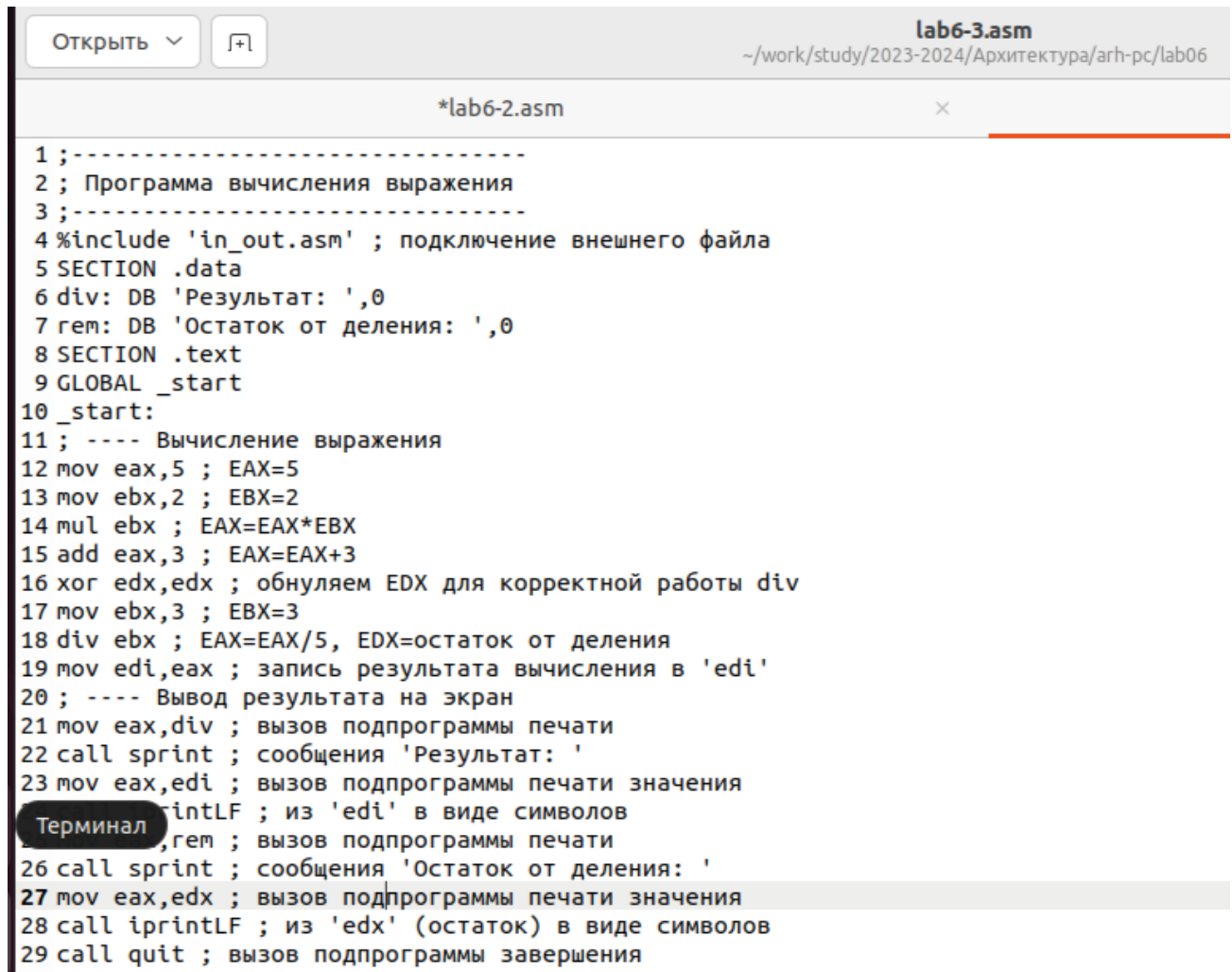


Figure 13: Создание файла

Создаю исполняемый файл и запускаю его. (рис. 20).

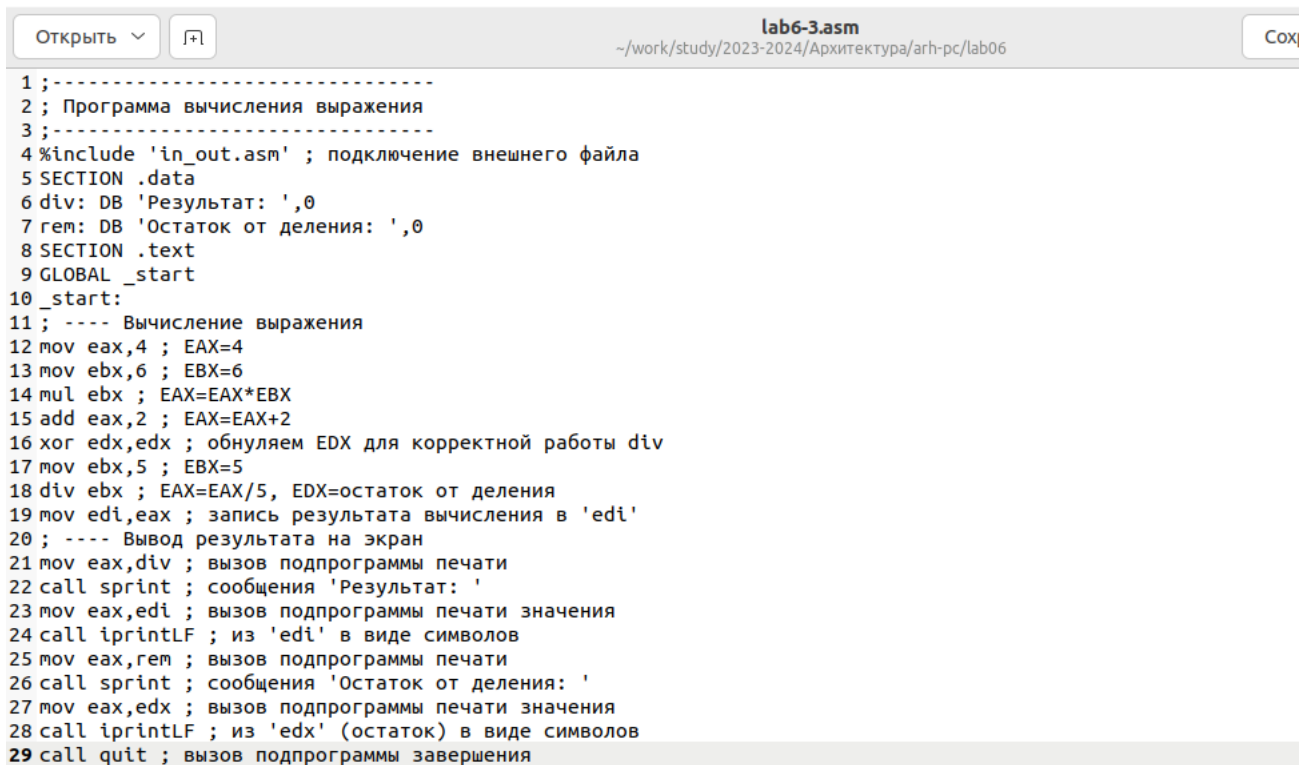
```

dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ nasm -f elf lab6-3.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$

```

Figure 14: Запуск исполняемого файла

Изменяю текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$, делая замену чисел в регистрах. (рис. 20).



```

Открыть  [F] lab6-3.asm ~/work/study/2023-2024/Архитектура/arch-pc/lab06 Сох
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,4 ; EAX=4
13 mov ebx,6 ; EBX=6
14 mul ebx ; EAX=EAX*EBX
15 add eax,2 ; EAX=EAX+2
16 xor edx,edx ; обнуляем EDX для корректной работы div
17 mov ebx,5 ; EBX=5
18 div ebx ; EAX=EAX/5, 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

```

Figure 15: Изменение текста программы

Создаю исполняемый файл и проверяю его работу. (рис. 20).

```

dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ touch lab6-3.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ nasm -f elf lab6-3.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$

```

Figure 16: Запуск исполняемого файла

Создаю файл variant.asm в каталоге ~/work/arch-pc/lab06 (рис. 20).

```

dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$ touch variant.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arch-pc/lab06$

```

Figure 17: Создание файла

Текст программы из листинга 6.4 ввожу в файл variant.asm, создаю исполняемый файл и запускаю его. Проверяю результат работы программы, вычислив номер варианта аналитически (ответ верный). (рис. 20).

```
dgavdadaev@dgavdadaev-laptop: ~/work/study/2023-2024/Архитектура/arh-pc/lab06$ nasm -f elf variant.asm
dgavdadaev@dgavdadaev-laptop: ~/work/study/2023-2024/Архитектура/arh-pc/lab06$ ld -m elf_i386 -o variant variant.o
dgavdadaev@dgavdadaev-laptop: ~/work/study/2023-2024/Архитектура/arh-pc/lab06$ ./variant
Введите № студенческого билета:
1032230169
Ваш вариант: 10
dgavdadaev@dgavdadaev-laptop: ~/work/study/2023-2024/Архитектура/arh-pc/lab06$
```

Figure 18: Результат работы кода

4.2.1 Ответы на вопросы по листингу 6.4

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```
mov eax,rem
```

```
call sprint
```

2. `mov ecx, x` - Используется, чтобы положить адрес вводимой строки `x` в регистр.

`mov edx, 80` - Используется для записи в регистр `edx` длины вводимой строки.

`call sread` - Используется для вызова подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры.

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:

```
xor edx,edx
```

```
mov ebx,20
```

```
div ebx
```

```
inc edx
```

5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`.

6. Инструкция “`inc edx`” увеличивает значение регистра `edx` на 1.

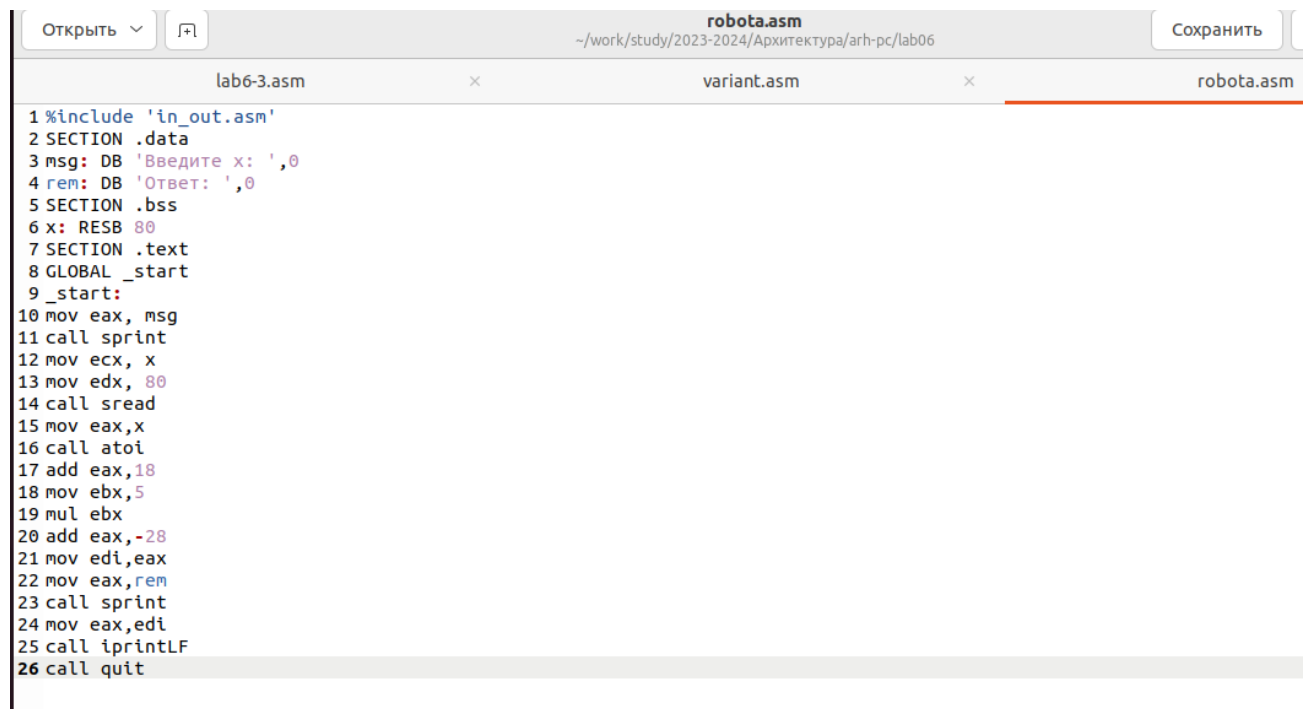
7. За вывод на экран результатов вычислений отвечают строки:

```
mov eax,edx
```

```
call iprintLF
```


4.3 Задание для самостоятельной работы

Вывод программы variant.asm показал, что мой номер варианта - 10, поэтому мне нужно написать программу (rabota.asm) для вычисления выражения $5(x + 18) - 28$ и проверить ее работу для значений $x_1 = 2$ и $x_2 = 3$. (рис. 20).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 rem: DB 'Ответ: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprint
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x
16 call atoi
17 add eax, 18
18 mov ebx, 5
19 mul ebx
20 add eax, -28
21 mov edi, eax
22 mov eax, rem
23 call sprint
24 mov eax, edi
25 call iprintlnLF
26 call quit
```

Figure 19: Создание программы

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg: DB 'Введите x:',0
```

```
rem: DB 'Ответ:',0
```

```
SECTION .bss
```

```
x: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

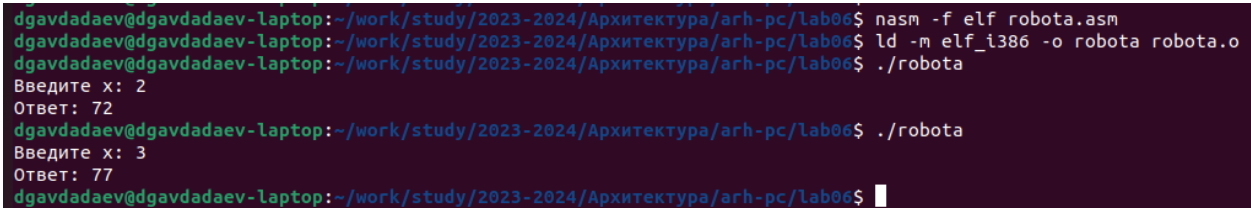
```
mov eax, msg
```

```
call sprint
```

```
mov ecx, x
```

```
mov edx, 80
call sread
mov eax,x
call atoi
add eax,18
mov ebx,5
mul ebx
add eax,-28
mov edi,eax
mov eax,rem
call sprint
mov eax,edi
call iprintLF
call quit
```

Создаю исполняемый файл и проверяю его работу. (рис. 20).



```
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ nasm -f elf robota.asm
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ ld -m elf_i386 -o robota robota.o
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ ./robota
Введите x: 2
Ответ: 72
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$ ./robota
Введите x: 3
Ответ: 77
dgavdadaev@dgavdadaev-laptop:~/work/study/2023-2024/Архитектура/arh-pc/lab06$
```

Figure 20: Результат работы кода

5 Выводы

С помощью данной лабораторной работы я освоил арифметические инструкции языка ассемблер NASM, что пригодится мне при выполнении последующих лабораторных работ.

6 Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.

5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).
1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: <https://www.gnu.org/software/bash/manual/>.
2. Newham C. [Learning the bash Shell: Unix Shell Programming](#). O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. [Bash Pocket Reference](#). O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.