

Лабораторная работа номер 6

Отчёт

Виноградова Мария Андреевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Символьные и численные данные в NASM	8
4.2	Выполнение арифметических операций в NASM	11
4.3	Ответы на вопросы по программе	14
4.4	Задание для самостоятельной работы	15
5	Выводы	17

Список иллюстраций

4.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code>	8
4.2	Заполняем файл	8
4.3	Запускаем файл и смотрим на его работу	9
4.4	Изменяем файл	9
4.5	Запускаем файл и смотрим на его работу	9
4.6	Создаем файл	9
4.7	Заполняем файл	10
4.8	Смотрим на работу программы	10
4.9	Изменяем файл	10
4.10	Смотрим на работу программы	10
4.11	Изменяем файл	11
4.12	Смотрим на работу программы	11
4.13	Создаем файл	11
4.14	Заполняем файл	12
4.15	Смотрим на результат работы программы	12
4.16	Редактируем файл	13
4.17	Смотрим на результат работы программы	13
4.18	Создаем файл	13
4.19	Заполняем файл	14
4.20	Проверяем результат работы программы	14
4.21	Создаем файл	15
4.22	Заполняем файл	16
4.23	Проверяем работу программы	16
4.24	Проверяем работу программы	16

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

Написать программы для вычисления арифметических выражений с неизвестной.

3 Теоретическое введение

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

Создаем каталог для программ которые потребуются нам в ходе выполнения ЛБ6, и в нем создаем файл (рис. fig. 4.1).

```
mavinogradova@fedora:~$ mkdir ~/work/arch-pc/lab06
mavinogradova@fedora:~$ cd ~/work/arch-pc/lab06
mavinogradova@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
mavinogradova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

Открываем файл в Midnight Commander и заполняем его как показано в листинге 6.1 (рис. fig. 4.2).



```
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab06/lab6-1.asm
#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 sprintf
call quit
```

Рис. 4.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. fig. 4.3).


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

Рис. 4.3: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и убираем кавычки с числовых значений (рис. fig. 4.4).

```
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab06/lab6-1.asm
#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
```

Рис. 4.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 4.5).

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

mavinogradova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.5: Запускаем файл и смотрим на его работу

Создаем новый файл в каталоге (рис. fig. 4.6).

```
mavinogradova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
mavinogradova@fedora:~/work/arch-pc/lab06$ mc
```

Рис. 4.6: Создаем файл

Заполняем файл как показано в листинге 6.2 (рис. fig. 4.7).



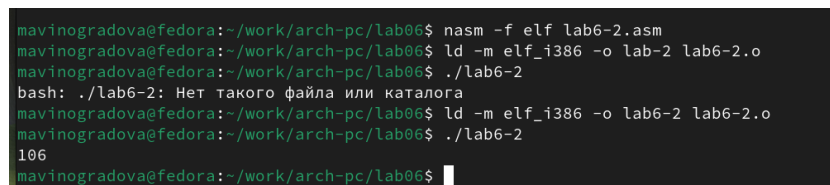
```
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 4.7: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. fig. 4.8).



```
mavinogradova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
mavinogradova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
mavinogradova@fedora:~/work/arch-pc/lab06$ ./lab6-2
bash: ./lab6-2: Нет такого файла или каталога
mavinogradova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
mavinogradova@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
mavinogradova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.8: Смотрим на работу программы

Снова открываем файл для редактирования и убираем кавычки с числовых значений (рис. fig. 4.9).



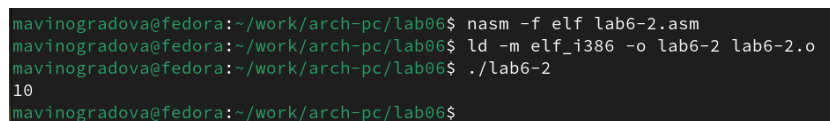
```
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 4.9: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 4.10).



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

Рис. 4.10: Смотрим на работу программы

Снова открываем файл для редактирования и меняем `iprintLF` на `iprint` (рис. fig. 4.11).



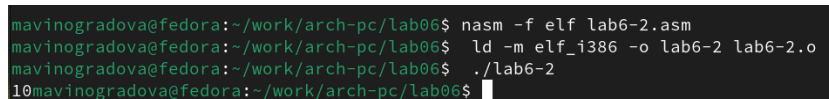
```
mavinogradova@fedora:~/work/arch-pc/lab06
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab06/lab6-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:

mov eax, 6
mov ebx, 4
add eax, ebx
call iprint

call quit
```

Рис. 4.11: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 4.12).



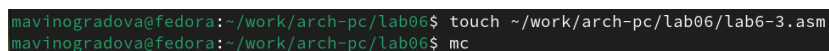
```
mavinogradova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
mavinogradova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
mavinogradova@fedora:~/work/arch-pc/lab06$ ./lab6-2
10mavinogradova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.12: Смотрим на работу программы

Вывод функций `iprintLF` и `iprint` отличаются тем, что `iprintLF` переносит вывод на новую строку.

4.2 Выполнение арифметических операций в NASM

Создаем новый файл в каталоге (рис. fig. 4.13).



```
mavinogradova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
mavinogradova@fedora:~/work/arch-pc/lab06$ mc
```

Рис. 4.13: Создаем файл

Открываем файл и редактируем его так как показано в листинге 6.3 (рис. fig. 4.14).

```
mavinogradova@fedora:~/work/arch-pc/lab06
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab06/lab6-3.asm
#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
```

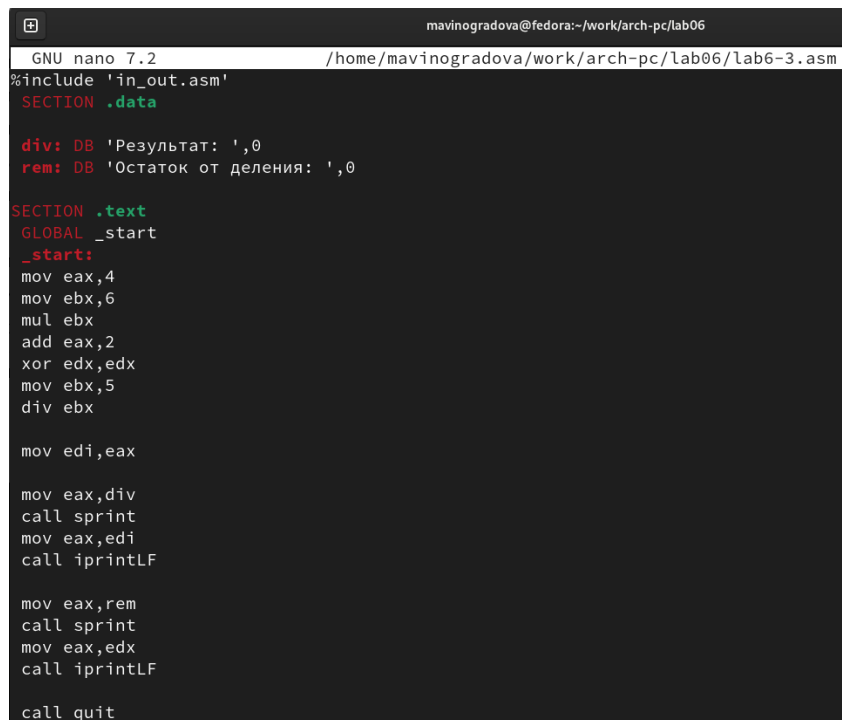
Рис. 4.14: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. fig. 4.15).

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

Рис. 4.15: Смотрим на результат работы программы

Открываем файл и редактируем так чтобы программа работала на вычисление выражения $f(x) = (4 \cdot x + 2)/5$ (рис. fig. 4.16).



```
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab06/lab6-3.asm
#include 'in_out.asm'
SECTION .data

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

SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx

mov edi,eax

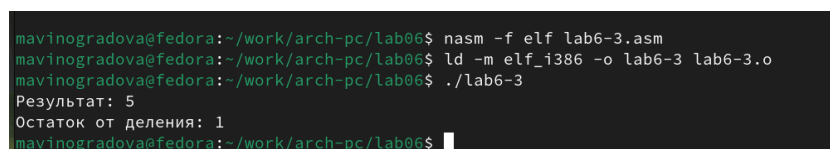
mov eax,div
call sprintf
mov eax,edi
call fprintf

mov eax,rem
call sprintf
mov eax,edx
call fprintf

call quit
```

Рис. 4.16: Редактируем файл

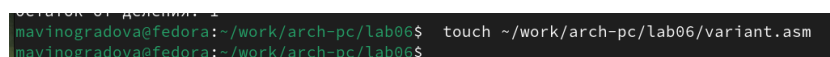
Компилируем файл и запускаем программу (рис. fig. 4.17).



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

Рис. 4.17: Смотрим на результат работы программы

Создаем новый файл в каталоге (рис. fig. 4.18).



```
Остаток от деления: 1
mavinogradova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
mavinogradova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.18: Создаем файл

Открываем файл и редактируем его так как показано в листинге 6.4 (рис. fig. 4.19).

```
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab06/variant.asm
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 sprintf

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 sprintf
mov eax, edx
call iprintf

call quit
```

Рис. 4.19: Заполняем файл

Компилируем файл и запускаем его (рис. fig. 4.20).

```
mavinogradova@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
mavinogradova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
mavinogradova@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132240691
Ваш вариант: 12
mavinogradova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.20: Проверяем результат работы программы

4.3 Ответы на вопросы по программе

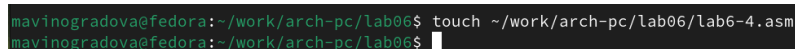
1. Строка “mov eax,rem” и строка “call sprintf” отвечают за вывод на экран сообщения ‘Ваш вариант:’.
2. Эти инструкции используются для чтения строки с вводом данных от пользователя. Начальный адрес строки сохраняется в регистре ecx, а количество символов в строке (максимальное количество символов, которое может быть считано) сохраняется в регистре edx. Затем вызывается процедура

sread, которая выполняет чтение строки.

3. Инструкция “call atoi” используется для преобразования строки в целое число. Она принимает адрес строки в регистре еах и возвращает полученное число в регистре еах.
4. Строка “xor edx,edx” обнуляет регистр edx перед выполнением деления. Строка “mov ebx,20” загружает значение 20 в регистр ebx. Строка “div ebx” выполняет деление регистра еах на значение регистра ebx с сохранением частного в регистре еах и остатка в регистре edx.
5. Остаток от деления записывается в регистр edx.
6. Инструкция “inc edx” используется для увеличения значения в регистре edx на 1. В данном случае, она увеличивает остаток от деления на 1.
7. Строка “mov еах,edx” передает значение остатка от деления в регистр еах. Строка “call iprintLF” вызывает процедуру iprintLF для вывода значения на экран вместе с переводом строки.

4.4 Задание для самостоятельной работы

Создаем новый файл в каталоге (рис. fig. 4.21).



```
mavinogradova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-4.asm
mavinogradova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.21: Создаем файл

Открываем файл и редактируем так чтобы программа работала на вычисление выражения под номером который мы получили в ходе выполнения работы $((8\pi - 6)/2)$ (рис. fig. 4.22).

```
mavinogradova@fedora:~/work/arch-pc/lab06
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab06/lab6-4.asm
#include 'in_out.asm'

SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',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
mov ebx, 8
mul ebx
add eax, -6
xor edx, edx
mov ebx, 2
div ebx
mov edi, eax
mov eax, rem
```

Рис. 4.22: Заполняем файл

Компилируем программу и проверяем её работу для первого значения x ($x=1$) (рис. fig. 4.23).

```
mavinogradova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
mavinogradova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
mavinogradova@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 1
```

Рис. 4.23: Проверяем работу программы

Компилируем программу и проверяем её работу для второго значения x ($x=5$) (рис. fig. 4.24).

```
mavinogradova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
mavinogradova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
mavinogradova@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 5
Результат: 17
```

Рис. 4.24: Проверяем работу программы

5 Выводы

Мы приобрели навыки создания исполнительных файлов для решения выражений и освоили арифметические инструкции в NASM.