

# **Отчёт по лабораторной работе №6**

Иванова Анастасия Сергеевна

# **Содержание**

<b>1 Цель работы</b>	<b>5</b>
<b>2 Порядок выполнения лабораторной работы</b>	<b>6</b>
<b>3 Выполнение заданий для самостоятельной работы</b>	<b>16</b>
<b>4 Вывод</b>	<b>18</b>

# **Список иллюстраций**

2.1	Создание каталога и файла . . . . .	6
2.2	Ввод программы . . . . .	6
2.3	Проверка работы программы . . . . .	7
2.4	Изменение программы . . . . .	8
2.5	Проверка работы программы . . . . .	8
2.6	Создание файла . . . . .	9
2.7	Ввод программы . . . . .	9
2.8	Проверка работы программы . . . . .	9
2.9	Изменение программы . . . . .	10
2.10	Проверка работы программы . . . . .	10
2.11	Изменение программы . . . . .	11
2.12	Запуск программы . . . . .	11
2.13	Создание файла . . . . .	11
2.14	Ввод программы . . . . .	12
2.15	Проверка работы программы . . . . .	12
2.16	Изменение программы . . . . .	13
2.17	Проверка работы программы . . . . .	13
2.18	Создание файла . . . . .	14
2.19	Ввод программы . . . . .	14
2.20	Проверка работы программы . . . . .	14
3.1	ВСоздание файла . . . . .	16
3.2	Ввод программы . . . . .	17
3.3	Проверка работы программы . . . . .	17

# **Список таблиц**

# **1 Цель работы**

Освоить арифметические инструкции языка ассемблера NASM.

## 2 Порядок выполнения лабораторной работы

1. Создадим каталог для программ лабораторной работы № 6, перейдем в него и создадим файл lab6-1.asm рис. 2.1:

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

Рисунок 2.1: Создание каталога и файла

2. Рассмотрим пример программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax. Введем в файл lab6-1.asm текст программы рис. 2.2:

```
GNU nano 7.2      /home/asivanova/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
```

Рисунок 2.2: Ввод программы

Создадим исполняемый файл и запустим его рис. 2.3:

```
asivanova1@anastasia:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
asivanova1@anastasia:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
asivanova1@anastasia:~/work/arch-pc/lab06$ ./lab6-1
j
```

Рисунок 2.3: Проверка работы программы

В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом является символ j. Команда add eax,ebx записывает в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j. З. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы рис. 2.4:

```
%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
```

Рисунок 2.4: Изменение программы

Создадим исполняемый файл и запустим его рис. 2.5:

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

Рисунок 2.5: Проверка работы программы

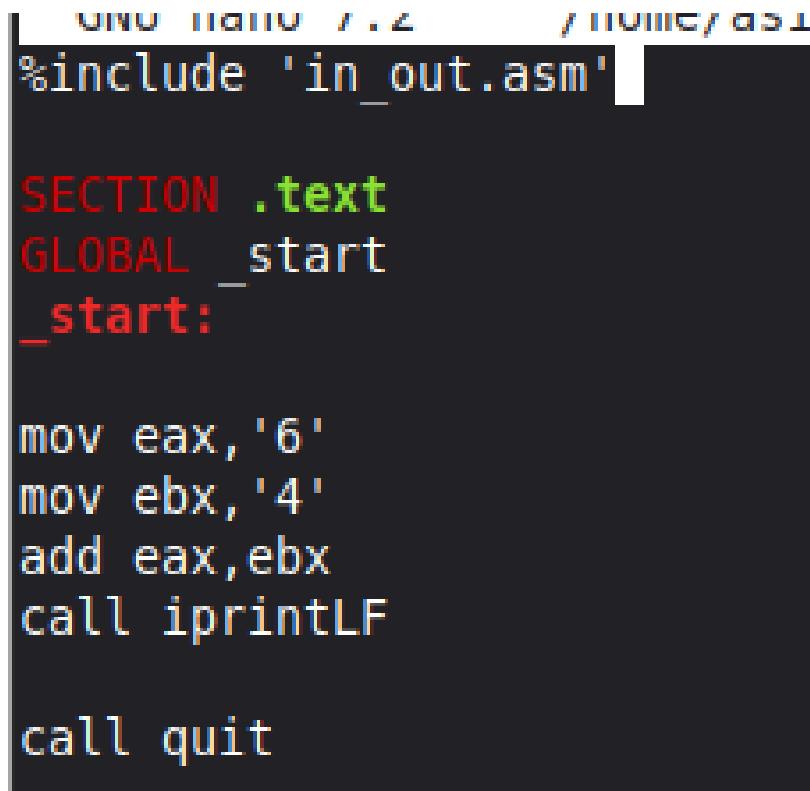
Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае должен выводится символ с кодом 10, но он не отобразился. 4. Для работы с числами в файле in\_out.asm реализованы подпрограммы для преоб-

разования ASCII символов в числа и обратно. Создадим файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введем в него текст программы рис. 2.6:

```
asivanova1@anastasia:~/work/arch-pc/lab06$ touch lab6-2.asm
```

Рисунок 2.6: Создание файла

рис. 2.7



```
%include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

    call quit
```

Рисунок 2.7: Ввод программы

Создадим исполняемый файл и запустим его рис. 2.8:

```
asivanova1@anastasia:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
asivanova1@anastasia:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
asivanova1@anastasia:~/work/arch-pc/lab06$ ./lab6-2
106
```

Рисунок 2.8: Проверка работы программы

В результате работы программы мы получаем число 106. 5. Аналогично предыдущему примеру изменим символы на числа. Заменим строки рис. 2.9:

```
%include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

    call quit
```

Рисунок 2.9: Изменение программы

Создадим исполняемый файл и запустим его рис. 2.10:

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

Рисунок 2.10: Проверка работы программы

В результате работы программы мы получаем число 10. Заменим функцию iprintLF на iprint. Создадим исполняемый файл и запустим его рис. 2.11:

```
UNO UNO / .4      / NUMS/a:  
%include 'in_out.asm'  
  
SECTION .text  
GLOBAL _start  
  
_start:  
  
    mov eax,6  
    mov ebx,4  
    add eax,ebx  
    call iprint  
  
    call quit
```

Рисунок 2.11: Изменение программы

рис. 2.12

```
asivanova1@anastasia:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm  
asivanova1@anastasia:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o  
asivanova1@anastasia:~/work/arch-pc/lab06$ ./lab6-2  
10asivanova1@anastasia:~/work/arch-pc/lab06$ █
```

Рисунок 2.12: Запуск программы

После замены функции число 10 выводится без переноса (не на отдельной строчке). 6. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения  $f(x) = (5 * 2 + 3)/3$ . Создадим файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 и напишем в нем программу рис. 2.13:

```
asivanova1@anastasia:~/work/arch-pc/lab06$ touch lab6-3.asm
```

Рисунок 2.13: Создание файла

рис. 2.14:

The screenshot shows a terminal window with assembly code. The code includes an include directive for 'in\_out.asm', data section with strings for result and remainder, and a text section with a global start label. The division logic uses eax for dividend, ebx for divisor, edx for remainder, and edi for result.

```
%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
```

Рисунок 2.14: Ввод программы

Создадим исполняемый файл и запустим его рис. 2.15:

The screenshot shows a terminal window with command-line steps to compile and run the assembly program. It includes the compilation command (nasm -f elf lab6-3.asm), linking command (ld -m elf\_i386 -o lab6-3 lab6-3.o), and the execution command (./lab6-3). The output shows the result and remainder of the division.

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

Рисунок 2.15: Проверка работы программы

Программа выводит результат - 4 и остаток от деления - 1. Изменим текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$  рис. 2.16:

```
GNU nano 7.2      /home/asivanova1
%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
```

Рисунок 2.16: Изменение программы

Создадим исполняемый файл и проверим его работу рис. 2.17:

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

Рисунок 2.17: Проверка работы программы

Программа выводит результат - 5 и остаток от деления - 1. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму: • вывести запрос на вве-

дение № студенческого билета • вычислить номер варианта по формуле: ( $S_n \bmod 20$ ) + 1, где  $S_n$  – номер студенческого билета (В данном случае  $a \bmod b$  – это остаток от деления  $a$  на  $b$ ). • вывести на экран номер варианта. Создадим файл variant.asm в каталоге ~/work/arch-pc/lab06 и напишем в нем программу рис. 2.18:

```
asivanova1@anastasia:~/work/arch-pc/lab06$ touch variant.asm
```

Рисунок 2.18: Создание файла

рис. 2.19

```
GNU nano 7.2      /home/asivanova1/work/arch-pc/
%include 'in_out.asm'

SECTION .data
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 sprintLF

    mov ecx, x
    mov edx, 80
    call sread
```

Рисунок 2.19: Ввод программы

Создадим исполняемый файл и запустим его рис. 2.20:

```
asivanova1@anastasia:~/work/arch-pc/lab06$ nasm -f elf variant.asm
asivanova1@anastasia:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
asivanova1@anastasia:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132250427
Ваш вариант: 8
```

Рисунок 2.20: Проверка работы программы

Результатом работы является ответ - 8 вариант. 8. Ответы на следующие вопросы: 1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения „Ваш вариант: „?: mov eax,rem call sprint 2. Для чего используется следующие инструкции? mov ecx, x - загрузка адреса буфера x в ECX mov edx, 80 - загрузка размера буфера EDX call sread - вызов функции чтения строки с клавиатуры 3. Для чего используется инструкция «call atoi»? - преобразование строки в число. 4. Какие строки листинга 6.4 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 EDX. 6. Для чего используется инструкция «inc edx»? - увеличение значения в регистре на 1. 7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?: mov eax,rem

call sprint

mov eax,edx

call iprintLF

### **3 Выполнение заданий для самостоятельной работы**

1. Напишем программу для вычисления выражения  $y = (11 + x) * 2 - 6$  рис. 3.1:

```
asivanova@anastasia:~/work/arch-pc/lab06$ touch prog.asm
```

Рисунок 3.1: В Создание файла

рис. 3.2:

```

GNU nano 7.2                               /home/asivanoval/work/arch-pc/lab06/prog.asm
%include 'in_out.asm'

SECTION .data
msg_yrav DB 'Вычисляем выражение: y = (11 + x)*2 - 6', 0
msg_x DB 'Введите значение x: ', 0
msg_y DB 'Результат: y = ', 0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:
    mov eax, msg_yrav
    call sprintLF

    mov eax, msg_x
    call sprint

    mov ecx, x
    mov edx, 10
    call sread

    mov eax, x
    call atoi

    add eax, 11
    mov ebx, 2
    mul ebx
    sub eax, 6

    mov edi, eax
    mov eax, msg_y
    call sprint
    mov eax, edi
    call iprintLF

    call quit

```

Рисунок 3.2: Ввод программы

- Создадим исполняемый файл и проверим его работу для значений  $x_1 = 1$  и  $x_2 = 9$  рис. 3.3:

```

asivanoval@anastasia:~/work/arch-pc/lab06$ nasm -f elf prog.asm
asivanoval@anastasia:~/work/arch-pc/lab06$ ld -m elf_i386 -o prog prog.o
asivanoval@anastasia:~/work/arch-pc/lab06$ ./prog
Вычисляем выражение: y = (11 + x)*2 - 6
Введите значение x: 1
Результат: y = 18
asivanoval@anastasia:~/work/arch-pc/lab06$ nasm -f elf prog.asm
asivanoval@anastasia:~/work/arch-pc/lab06$ ld -m elf_i386 -o prog prog.o
asivanoval@anastasia:~/work/arch-pc/lab06$ ./prog
Вычисляем выражение: y = (11 + x)*2 - 6
Введите значение x: 10
Результат: y = 36

```

Рисунок 3.3: Проверка работы программы

## **4 Вывод**

Мы освоили арифметические инструкции языка ассемблера NASM.