

Отчет по лабораторной работе №6

Архитектура компьютеров и операционные системы

Никита Сергеевич Кокшаров

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
2.1	Символьные и численные данные в NASM	5
2.2	Выполнение арифметических операций в NASM	8
2.3	Ответы на вопросы	11
3	Задание для самостоятельной работы	14
4	Выводы	16

Список иллюстраций

2.1	Создание lab6-1.asm	5
2.2	Код в lab6-1.asm	5
2.3	Запуск lab6-1	6
2.4	Изменение кода в lab6-1.asm	6
2.5	Запуск измененного lab6-1	6
2.6	Код в lab6-2.asm	7
2.7	Запуск lab6-2	7
2.8	Изменение кода в lab6-2.asm	7
2.9	Запуск измененного lab6-2	7
2.10	Изменение iprintLF на iprint в lab6-3	8
2.11	Запуск lab6-2 с iprint	8
2.12	Код в lab6-3.asm	9
2.13	Запуск lab6-3	9
2.14	Измененный код lab6-3.asm	10
2.15	Запуск измененного lab6-3	10
2.16	Код в variant.asm	11
2.17	Запуск variant.asm	11
3.1	Код в function.asm	14
3.2	Запуск function	15

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

2.1 Символьные и численные данные в NASM

Создаю файл lab6-1.asm в новой директории lab06 (рис. 2.1).

```
nkshrv@DESKTOP-V695CJT:~$ mkdir ~/work/arch-pc/lab06
nkshrv@DESKTOP-V695CJT:~$ cd ~/work/arch-pc/lab06
nkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ touch lab6-1.asm
nkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ls
lab6-1.asm
nkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$
```

Рис. 2.1: Создание lab6-1.asm

Пишу код программы из листинга 6.1 (рис. 2.2)

```
mc [nkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06]
/home/nkshrv/work~/lab06/lab6-1.asm [----] 13 L: [ 1+16 17/ 17] *(216 / 216b) <EOF> [*][X]
#include "in_out.asm"

SECTION .bss
buf1: <-> RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, '0'
mov ebx, '4'
add eax, ebx
mov [buf1], eax
mov ecx, buf1
call sprintf
call quit
```

Рис. 2.2: Код в lab6-1.asm

Создаю исполняемый файл и запускаю его (рис. 2.3)

```
nkksrv@DESKTOP-V695CJT: ~/work/arch-pc/lab06
nkksrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
nkksrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
nkksrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o
nkksrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ./lab6-1
j
nkksrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$
```

Рис. 2.3: Запуск lab6-1

Изменяю код программы (рис. 2.4)

```
mc [nkksrv@DESKTOP-V695CJT:~/work/arch-pc/lab06
/home/nkksrv/work~/lab06/lab6-1.asm [-M--] 13 L: [ 1+16 17/ 17] *(212 / 212b) <EOF> [*][X]
#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 cap, buf1
call sprintLF

call quit
```

Рис. 2.4: Изменение кода в lab6-1.asm

Создаю исполняемый файл и запускаю его (рис. 2.5)

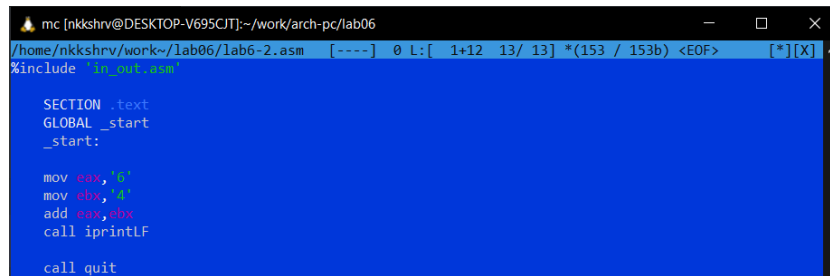
```
nkksrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
nkksrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
nkksrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ./lab6-1

nkksrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$
```

Рис. 2.5: Запуск измененного lab6-1

Во втором случае выводится символ с кодом 10. В соответствии с таблицей ASCII таким символом является STX (Start of Text). Он не отображается.

Создаю lab6-2.asm. Пишу код программы из листинга 6.2 (рис. 2.6)



```

/home/nkkshrv/work~/lab06/lab6-2.asm  [----]  0 L: [ 1+12 13/ 13] *(153 / 153b) <EOF>  [*][X] ^
#include "in_out.asm"

SECTION .text
GLOBAL _start
_start:

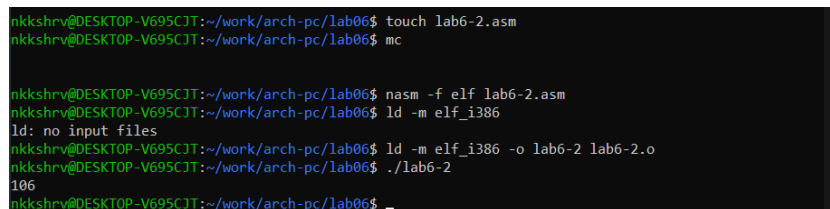
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF

call quit

```

Рис. 2.6: Код в lab6-2.asm

Создаю исполняемый файл и запускаю его (рис. 2.7)



```

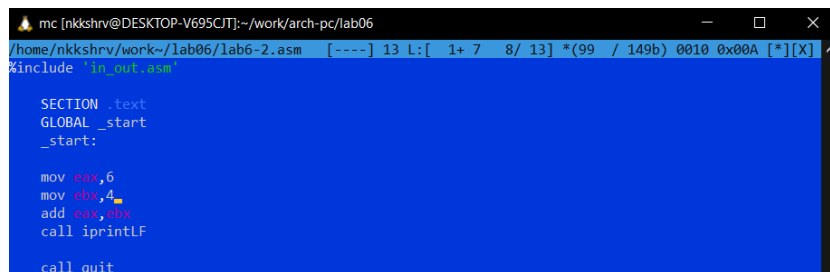
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ touch lab6-2.asm
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ mc

nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ld -m elf_i386
ld: no input files
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ./lab6-2
106
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$

```

Рис. 2.7: Запуск lab6-2

Изменяю код программы (рис. 2.8)



```

/home/nkkshrv/work~/lab06/lab6-2.asm  [----] 13 L: [ 1+ 7 8/ 13] *(99 / 149b) 0010 0x00A [*][X] ^
#include "in_out.asm"

SECTION .text
GLOBAL _start
_start:

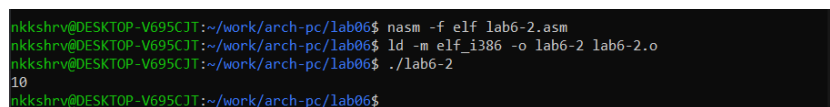
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF

call quit

```

Рис. 2.8: Изменение кода в lab6-2.asm

Создаю исполняемый файл и запускаю его (рис. 2.9)



```

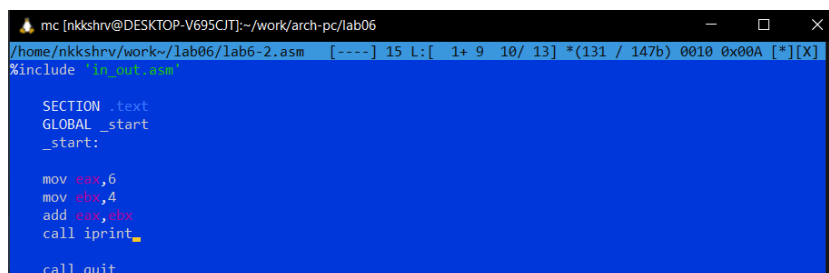
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ./lab6-2
10
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$

```

Рис. 2.9: Запуск измененного lab6-2

Результат выполнения программы поменялся: вывод 106 сменился на вывод 10.

Меняю вызов подпрограммы `iprintLF` на `iprint` (рис. 2.10)



```
mc [nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06]
/home/nkkshrv/work/~lab06/lab6-2.asm [----] 15 L:[ 1+ 9 10/ 13] *(131 / 147b) 0010 0x00A [*][X]
#include "in_out.asm"

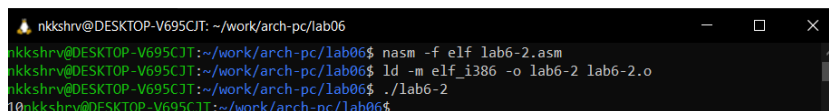
SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 2.10: Изменение `iprintLF` на `iprint` в lab6-3

Создаю исполняемый файл и запускаю его (рис. 2.11)



```
nkkshrv@DESKTOP-V695CJT: ~/work/arch-pc/lab06
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ./lab6-2
10nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$
```

Рис. 2.11: Запуск lab6-2 с `iprint`

2.2 Выполнение арифметических операций в NASM

Пишу код программы из листинга 6.3 (рис. 2.12)


```

mc [nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06
/home/nkkshrv/work~/lab06/lab6-3.asm [----] 13 L: [ 1+31 32/ 32] *(443 / 443b) <EOF> [*][X] ^
#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 ecx,div
call sprint
mov eax,edi
call iprintLF

mov ecx,rem
call sprint
mov ecx,ebx
call iprintLF

call quit

```

Рис. 2.12: Код в lab6-3.asm

Создаю исполняемый файл и запускаю его (рис. 2.13)

```

nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$

```

Рис. 2.13: Запуск lab6-3

Меняю код программы для вычисления значения выражения $f(x) = (4 * 6 + 2) / 5$ (рис. 2.14)

```

mc [nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06
/home/nkkshrv/work~/lab06/lab6-3.asm [-M--] 13 L: [ 1+16 17/ 32] *(265 / 443b) 0010 0x00A [*][X]
%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 sprint
mov eax,edi
call iprintLF

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

call quit

```

Рис. 2.14: Измененный код lab6-3.asm

Создаю исполняемый файл и запускаю его (рис. 2.15)

```

nkkshrv@DESKTOP-V695CJT: ~/work/arch-pc/lab06
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ mc
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$

```

Рис. 2.15: Запуск измененного lab6-3

Результат верный.

Пишу код программы из листинга 6.4 (рис. 2.16)

```

mc [nkkshrv@DESKTOP-V695CJT]:~/work/arch-pc/lab06
/home/nkkshrv/work~lab06/variant.asm  [----] 13 L: [ 1+33 34/ 34] *(494 / 494b) <EOF>  [*][X] ^
%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 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. 2.16: Код в variant.asm

Создаю исполняемый файл и запускаю его (рис. 2.17)

```

nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ touch variant.asm
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ mc

nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ nasm -f elf variant.asm
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132236037
Ваш вариант: 18
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$

```

Рис. 2.17: Запуск variant.asm

2.3 Ответы на вопросы

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант:?'?

Ответ:

```
mov eax, rem
```

```
call sprint
```

2. Для чего используются следующие инструкции?

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

Ответ: Инструкция `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. Для чего используется инструкция “`call atoi`”?

Ответ: С помощью “`call atoi`” мы вызываем функцию `atoi`, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax` (перед вызовом `atoi` в регистр `eax` необходимо записать число).

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

Ответ:

```
xor edx,edx  
mov ebx,20  
div ebx  
inc edx
```

5. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”?

Ответ:

Остаток записывается в регистр `edx`.

6. Для чего используется инструкция “`inc edx`”?

Ответ:

Для того, чтобы инкрементировать значение в регистре `edx`.

7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычисления?

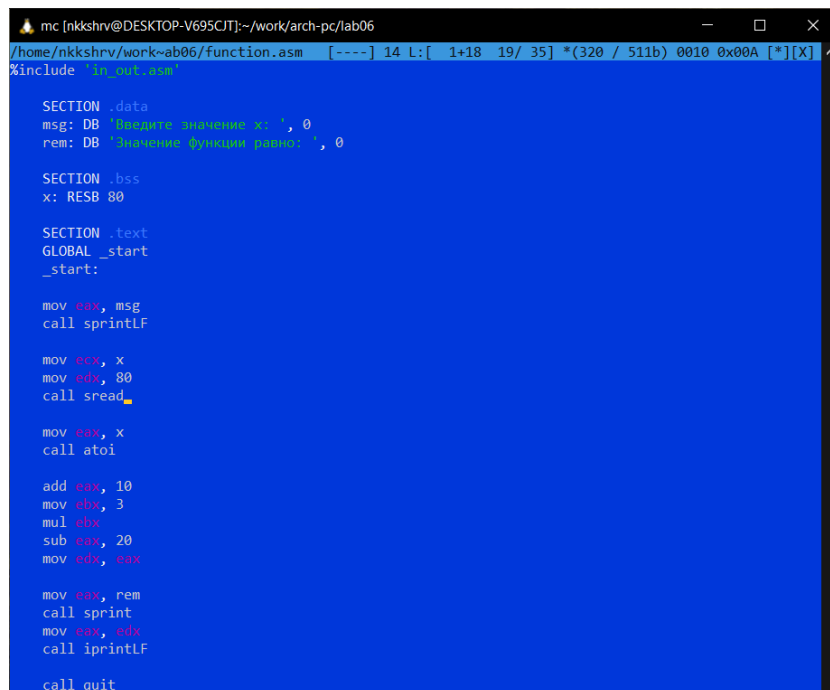
Ответ:

```
mov eax,edx  
call iprintLF
```

3 Задание для самостоятельной работы

Мой вариант — номер 18.

Пишу для вычисления значения данной функции $f(x) = 3(x + 10) - 20$ (рис. 3.1)



```
mc [nkshrv@DESKTOP-V695CJT]:~/work/arch-pc/lab06
/home/nkshrv/work~ab06/function.asm [----] 14 L: [ 1+18 19/ 35] *(320 / 511b) 0010 0x00A [*][X]
#include "in_out.asm"

SECTION .data
msg: DB 'Введите значение x: ', 0
rem: DB 'Значение функции равно: ', 0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov ecx, msg
call sprintf

mov ecx, x
mov edx, 80
call sread

mov ecx, x
call atoi

add ecx, 10
mov ebx, 3
mul ebx
sub ecx, 20
mov edx, ecx

mov ecx, rem
call sprintf
mov ecx, edx
call iprintf

call quit
```

Рис. 3.1: Код в function.asm

Создаю исполняемый файл и запускаю его. Ввожу данные значения x для проверки. Получаю верные ответы (рис. 3.2)

```
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ nasm -f elf function.asm
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ld -m elf_i386 -o function function.o
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ./function
Введите значение x:
1
Значение функции равно: 13
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$ ./function
Введите значение x:
5
Значение функции равно: 25
nkkshrv@DESKTOP-V695CJT:~/work/arch-pc/lab06$
```

Рис. 3.2: Запуск function

4 Выводы

При выполнении данной лабораторной работы я освоил арифметические инструкции языка ассемблера NASM.