

Отчёт по лабораторной работе №6

Арифметические операции в NASM.

Попутников Егор Сергеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выполнение самостоятельной работы	15
5	Выводы	17

Список иллюстраций

3.1	Создание файла	7
3.2	Текст из листинга	8
3.3	Создание исполняемого файла	8
3.4	Изменение текста листинга	9
3.5	Создание исполняемого файла	9
3.6	Преобразование текста из листинга	10
3.7	Создание исполняемого файла	10
3.8	Преобразование текста из листинга	11
3.9	Создание файла	11
3.10	Создание исполняемого файла	12
3.11	Изменение текста из листинга 6.3	12
3.12	Запись текста из листинга 6.4	13
3.13	Создание исполняемого файла	13
4.1	Выполнение самостоятельной работы	16

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

Написать программу вычисления выражения $y=f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений x_1 и x_2 из 6.3.

3 Выполнение лабораторной работы

1. Создадим каталог для программ лабораторной работы № 6, перейдем в него и создадим файл lab6-1.asm:(3.1)

```
egor@espoputnikov-dk3n56:~$ cd ~/work/arch-pc
egor@espoputnikov-dk3n56:~/work/arch-pc$ mkdir ~/work/arch-pc/lab06
egor@espoputnikov-dk3n56:~/work/arch-pc$ cd ~/work/arch-pc/lab06
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ touch lab6-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ mc
```

Рис. 3.1: Создание файла

2. Введём в файл lab6-1.asm текст программы из листинга 6.1.(3.2).

```

#include 'in_out.asm'

SECTION .bss
buf1:      RESB 80

SECTION .text
GLOBAL _start
_start:

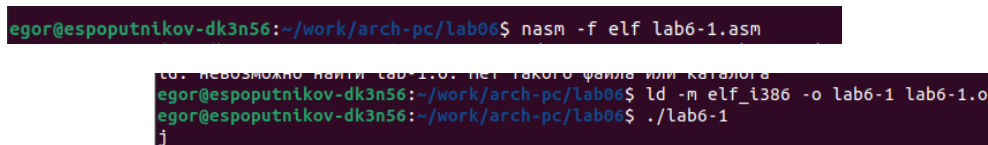
    mov     eax, '6'
    mov     ebx, '4'
    add     eax, ebx
    mov     [buf1], eax
    mov     eax, buf1
    call    sprintfLF

    call    quit

```

Рис. 3.2: Текст из листинга

3. Создадим исполняемый файл и запустим его.(3.3).



```

egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ./lab6-1
j

```

Рис. 3.3: Создание исполняемого файла

4. Далее изменим текст программы и вместо символов, запишем в регистры числа. Ис- правим текст программы (Листинг 6.1) следующим образом: заменим строки: `mov eax, '6'` `mov ebx, '4'` на строки `mov eax, 6` `mov ebx, 4` (3.4).


```
%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 sprintf
call quit
```

Рис. 3.4: Изменение текста листинга

5. Создадим исполняемый файл и запустим его.(3.5).

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ./lab6-1
```

Рис. 3.5: Создание исполняемого файла

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Число 10 в таблице ASCII соответствует символу переноса строки, следовательно, на экран ничего не выводится.

6. Преобразуем текст программы из Листинга 6.1 с использованием функций из файла in_out.asm. Создадим файл lab6-2.asm исправим в нём текст из листинга и создадим исполняемый файл.(3.6)

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

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

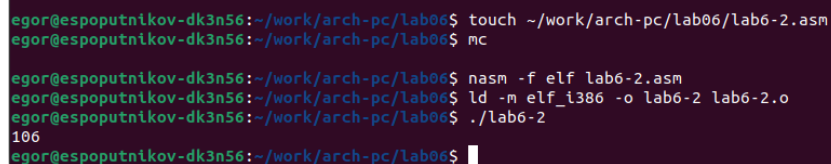
```
mov    eax,'6'
```

```
mov    ebx,'4'
```

```
add    eax,ebx
```

```
call   iprintLF
```

```
call   quit
```

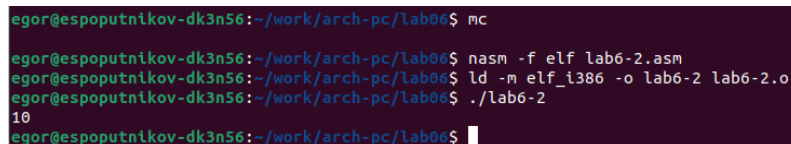


```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ mc

egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ./lab6-2
106
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$
```

Рис. 3.6: Преобразование текста из листинга

7. Аналогично предыдущему примеру изменим символы на числа. Заменяем строки: `mov eax,'6'` `mov ebx,'4'` на строки `mov eax,6` `mov ebx,4` Создадим исполняемый файл и запустим его. (3.7)



```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ mc

egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ./lab6-2
10
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$
```

Рис. 3.7: Создание исполняемого файла

В результате выполнения программы был получен результат 10.

8. Заменяем функцию `iprintLF` на `iprint`. Создадим исполняемый файл и запустим его.(3.8)

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

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ mc
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ./lab6-2
10egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$
```

Рис. 3.8: Преобразование текста из листинга

Функции отличаются тем, что `iprintLF` переводит написанный текст на следующую строку, а `iprint` нет.

9. Создадим файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06`(3.9)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
```

Рис. 3.9: Создание файла

10. Запишем текст из листинга 6.3, создадим исполняемый файл и запустим его.(3.10)

```

;-----
; Программа вычисления выражения
;-----
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран

```

```

egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$

```

Рис. 3.10: Создание исполняемого файла

11. Изменим текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$.
Создадим исполняемый файл и проверим его работу.(3.11)

```

egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$

```

Рис. 3.11: Изменение текста из листинга 6.3

12. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:
 - вывести запрос на введение № 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. В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символ-ном виде и для корректной работы арифметических операций в NASM символы

необходимо преобразовать в числа. Для этого может быть использована функция `atoi` из файла `in_out.asm`. Создадим файл `variant.asm` в каталоге `~/work/arch-pc/lab06`:

```
touch ~/work/arch-pc/lab06/variant.asm
```

Запишем в него текст из листинга 6.4: (3.12)

```
;-----  
; Программа вычисления варианта  
;-----  
%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 ; вызов подпрограммы преобразования  
call atoi ; ASCII кода в число, 'eax=x'  
xor edx, edx
```

Рис. 3.12: Запись текста из листинга 6.4

Создадим исполняемый файл и проверим работу программы.(3.13)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ nasm -f elf variant.asm  
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o  
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ./variant  
Введите № студенческого билета:  
1132231843  
Ваш вариант: 4  
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$
```

Рис. 3.13: Создание исполняемого файла

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант: '? За вывод сообщения на экран отвечают строки: `rem: DB 'Ваш вариант:',0` `mov eax,rem` `call sprint`
2. Для чего используются следующие инструкции? `mov ecx, x` `mov edx, 80` `call sread` Добавление в регистр `ecx` `x`, добавление в регистр `edx` `80`, чтение с консоли

3. Для чего используется инструкция “call atoi”? Преобразование символа 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”? Прибавление 1 к регистру `edx`.
7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений? `call iprintLF`

4 Выполнение самостоятельной работы

Напишем программу вычисления выражения $y = 4/3 \cdot (x - 1) + 5$. Создадим исполняемый файл и проверим его работу для значений x_1 и $x_2(4,10)$ (4.1)

```
1  %include 'in_out.asm'
2  SECTION .data
3  msg: DB 'x = ',0
4  rem: DB 'y = ',0
5  SECTION .bss
6  x: RESB 80
7  y: RESB 80
8  SECTION .text
9  GLOBAL _start
10 _start:
11 ;4/3(x-1)+5
12 mov eax, msg
13 call sprintf
14
15 mov ecx, x
16 mov edx, 80
17 call sread
18
19 mov eax,x ; вызов подпрограммы преобразования
20 call atoi ; ASCII кода в число, `eax=x`
21
22 xor edx,edx
23 dec eax
24 mov ebx,4
25 mul ebx
26 mov ebx,3
27 div ebx
28 add eax,5
29 mov ebx,eax
30
31 mov eax,rem
32 call sprintf
33 mov eax,ebx
34 call iprintLF
35 call quit
```

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ nasm -f elf funk.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ld -m elf_i386 -o funk funk.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ./funk
x =
4
y = 9
```

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab06$ ./funk  
x =  
10  
y = 17
```

Рис. 4.1: Выполнение самостоятельной работы

5 Выводы

Я освоил арифметические инструкции языка ассемблера NASM.