

Лабораторная работа №7

Дисциплина: Архитектура компьютера

Апареев Дмитрий Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Ответы на вопросы	13
4	Задания для самостоятельной работы	15
5	Вывод	17

Список иллюстраций

[illegible]

Список таблиц

1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

3 Выполнение лабораторной работы

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №7 (рис. 3.1). Перехожу в созданный каталог с помощью утилиты `cd` (рис. 3.2)

```
[daApareev@fedora ~]$ mkdir ~/work/study/2022-2023/"Архитектура компьютеров"/arh-pc/lab07
```

Рис. 3.1: 1

```
[daApareev@fedora ~]$ cd ~/work/study/2022-2023/"Архитектура компьютеров"/arh-pc/lab07
```

Рис. 3.2: 1.1

С помощью утилиты `touch` создаю файл `lab7-1.asm` (рис. 3.3).

```
[daApareev@fedora lab07]$ touch lab7-1.asm  
[daApareev@fedora lab07]$ ls  
lab7-1.asm
```

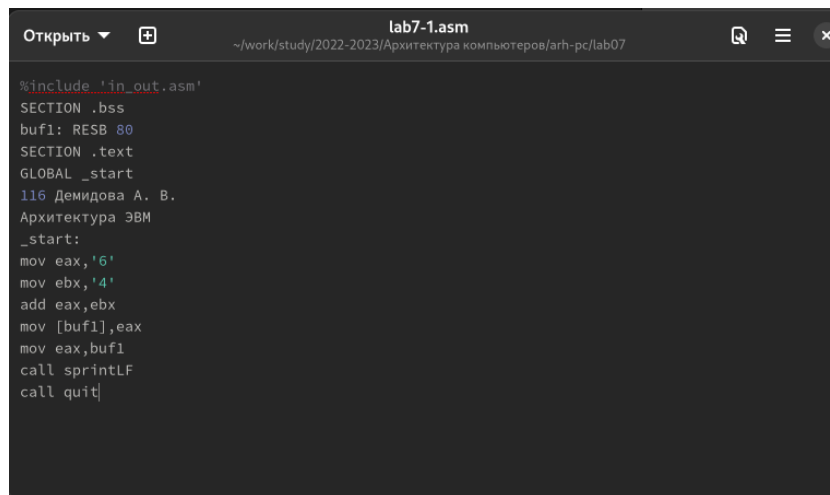
Рис. 3.3: 2

Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp` (рис. 3.4).

```
[daApareev@fedora lab07]$ cp ~/Загрузки/in_out.asm in_out.asm  
[daApareev@fedora lab07]$ ls  
in_out.asm lab7-1.asm
```

Рис. 3.4: 3

Открываю созданный файл `lab7-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 3.5).

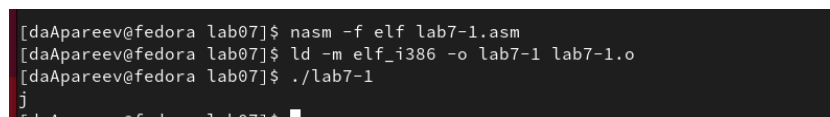


```
Открыть ▾ + lab7-1.asm
~/work/study/2022-2023/Архитектура компьютеров/arh-pc/lab07

%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
116 Демидова А. В.
Архитектура ЭВМ
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
mov [buf1], eax
mov eax, buf1
call printf
call quit
```

Рис. 3.5: 4

Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6. (рис. 3.6)



```
[daApareev@fedora lab07]$ nasm -f elf lab7-1.asm
[daApareev@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[daApareev@fedora lab07]$ ./lab7-1
j
[daApareev@fedora lab07]$
```

Рис. 3.6: 5

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 3.7).



```
Открыть ▾ + lab7-1.asm
~/work/study/2022-2023/Архитектура компьютеров/arh-pc/lab07

%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 printf
call quit
```

Рис. 3.7: 6

Создаю новый исполняемый файл программы и запускаю его. Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.(рис. 3.8)


```
[daApareev@fedora lab07]$ nasm -f elf lab7-1.asm
[daApareev@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[daApareev@fedora lab07]$ ./lab7-1
```

Рис. 3.8: 7

Создаю новый файл lab7-2.asm с помощью утилиты touch (рис. 3.9).

```
[daApareev@fedora lab07]$ touch lab7-2.asm
```

Рис. 3.9: 8

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. ??). 

Создаю и запускаю исполняемый файл lab7-2. Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”. (рис. 3.10).

```
[daApareev@fedora lab07]$ nasm -f elf lab7-2.asm
[daApareev@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[daApareev@fedora lab07]$ ./lab7-2
106
```

Рис. 3.10: 10

Заменяю в тексте программы в файле lab7-2.asm символы “6” и “4” на числа 6 и 4 (рис. 3.11).



```
Открыть ▾ + lab7-2.asm
~/work/study/2022-2023/Архитектура компьютеров/arh-pc/lab07

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

Рис. 3.11: 11

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10. (рис. 3.12)

```
[daApareev@fedora lab07]$ nasm -f elf lab7-2.asm
[daApareev@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[daApareev@fedora lab07]$ ./lab7-2
10
```

Рис. 3.12: 12

Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. 3.13).



```
Открыть ▾ + lab7-2.asm
~/work/study/2022-2023/Архитектура компьютеров/arh-pc/lab07

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

Рис. 3.13: 13

Создаю и запускаю новый исполняемый файл. Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией `iprintLF`, а `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF`. (рис. 3.14).

```
[daApareev@fedora lab07]$ nasm -f elf lab7-2.asm
[daApareev@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[daApareev@fedora lab07]$ ./lab7-2
10[daApareev@fedora lab07]$
```

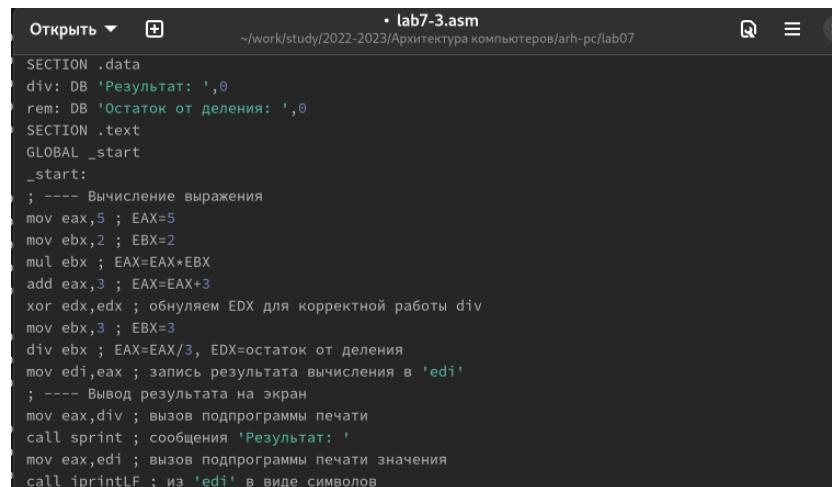
Рис. 3.14: 14

Создаю файл `lab7-3.asm` с помощью утилиты `touch` (рис. 3.15).

```
[daApareev@fedora lab07]$ touch lab7-3.asm
```

Рис. 3.15: 15

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 3.16).

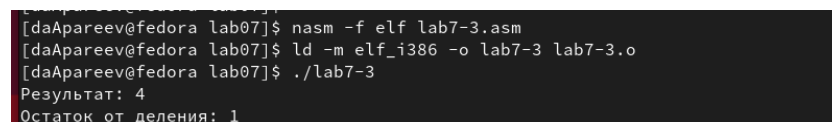


```
Открыть ▾ + lab7-3.asm
~/work/study/2022-2023/Архитектура компьютеров/ah-pc/lab07

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
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
```

Рис. 3.16: 16

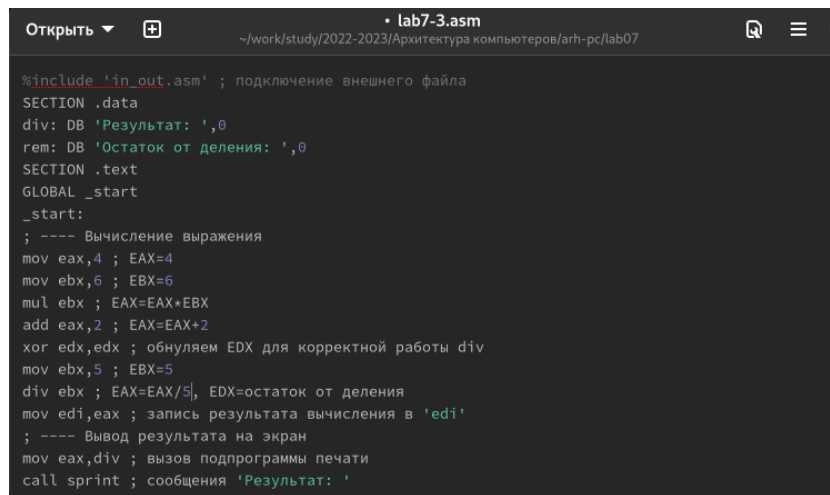
Создаю исполняемый файл и запускаю его (рис. 3.17).



```
[daApareev@fedora lab07]$ nasm -f elf lab7-3.asm
[daApareev@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[daApareev@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
```

Рис. 3.17: 17

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 3.18).



```

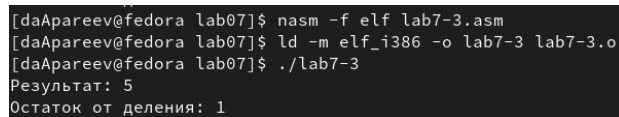
Открыть ▾ + • lab7-3.asm
~/work/study/2022-2023/Архитектура компьютеров/arh-pc/lab07

%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,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '

```

Рис. 3.18: 18

Создаю и запускаю новый исполняемый файл (рис. 3.19).



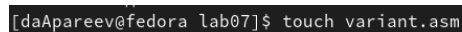
```

[daApareev@fedora lab07]$ nasm -f elf lab7-3.asm
[daApareev@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[daApareev@fedora lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1

```

Рис. 3.19: 19

Создаю файл variant.asm с помощью утилиты touch (рис. 3.20).



```

[daApareev@fedora lab07]$ touch variant.asm

```

Рис. 3.20: 20

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 3.21).

```

Открыть ▾ + variant.asm
~/work/study/2022-2023/Архитектура компьютеров/arh-pc/lab07
#include "in_out.asm"
SECTION .data
msg: DB 'Введите No 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
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
xor edx, edx
mov ebx, 20
div ebx
inc edx

```

Рис. 3.21: 21

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант 6. (рис. 3.22)

```

[daApareev@fedora lab07]$ nasm -f elf variant.asm
[daApareev@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[daApareev@fedora lab07]$ ./variant
Введите No студенческого билета:
1132226445
Ваш вариант: 6

```

Рис. 3.22: 22

3.1 Ответы на вопросы


1. За вывод сообщения “Ваш вариант” отвечают строки кода: `mov eax, rem` `call sprint`
2. 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. `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` ; обнуление `edx` для корректной работы `div` `mov ebx,20` ; `ebx = 20` `div ebx` ; `eax = eax/20`, `edx` - остаток от деления `inc edx` ; `edx = edx + 1`
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`

Открываю файл `report.md` с помощью любого текстового редактора `gedit`. Компилирую файл с отчетом. Загружаю отчет на GitHub.

4 Задания для самостоятельной работы

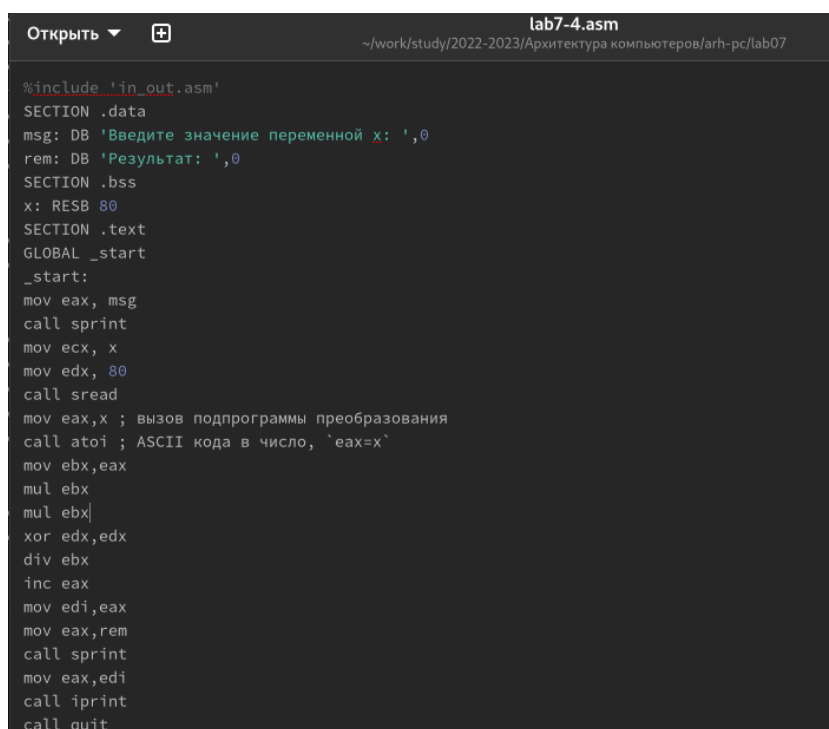
Создаю файл lab7-4.asm с помощью утилиты touch (рис. 4.1).



```
[daApareev@fedora lab07]$ touch lab7-4.asm
```

Рис. 4.1: 23

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $x^3 / 2 + 1$ (рис. 4.2).



```
lab7-4.asm
~/work/study/2022-2023/Архитектура компьютеров/arh-pc/lab07

#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 ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
mov ebx, eax
mul ebx
mul ebx
xor edx, edx
div ebx
inc eax
mov edi, eax
mov eax, rem
call sprint
mov eax, edi
call iprint
call quit
```

Рис. 4.2: 24

Создаю и запускаю исполняемый файл (рис. 4.3)

```
Результат: 17[daApareev@fedora lab07]$ nasm -f elf lab7-4.asm  
[daApareev@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o  
[daApareev@fedora lab07]$ ./lab7-4  
Введите значение переменной x: 2  
Результат: 5[daApareev@fedora lab07]$
```

Рис. 4.3: 25

Добавляю файлы в git , Сохраняю файлы в git , Отправляю файлы на сервер

5 Вывод

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