

Отчет по лабораторной работе №7

Дисциплина архитектура компьютера

Извекова Мария Петровна

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	13
4.3	Ответы на вопросы по программе	17
4.4	Самостоятельная работа	17
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	рис. 1	8
4.2	рис. 2	9
4.3	рис. 3	9
4.4	рис. 4	10
4.5	рис. 5	10
4.6	рис. 6	11
4.7	рис. 7	11
4.8	рис. 8	12
4.9	рис. 9	12
4.10	рис. 10	12
4.11	рис. 11	13
4.12	рис. 12	13
4.13	рис. 13	14
4.14	рис. 14	14
4.15	рис. 15	15
4.16	рис. 16	15
4.17	рис. 17	16
4.18	рис. 18	16
4.19	рис. 19	18
4.20	рис. 20	18
4.21	рис. 21	18
4.22	рис. 22	18

Список таблиц

1 Цель 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 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа адресации:

- Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`.
- Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`.
- Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

4 Выполнение лабораторной работы

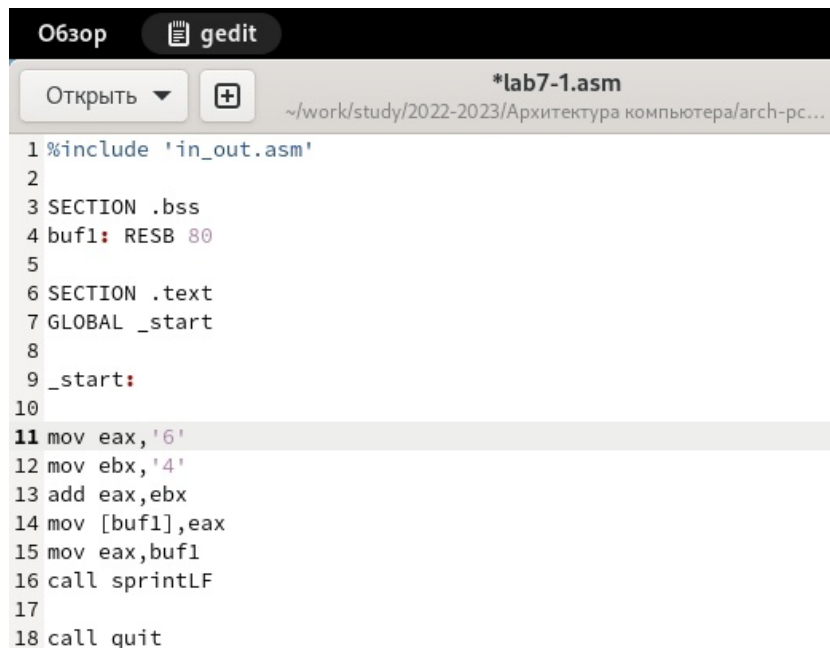
4.1 Символьные и численные данные в NASM

1. Переходим в каталог для программ Лабораторной работы №7, создаем файл lab7-1.asm и с помощью функции gedit открываем этот файл для редактирования (рис. 4.1)

```
[marie@fedora ~]$ cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab07  
[marie@fedora lab07]$ touch lab7-1.asm  
[marie@fedora lab07]$ gedit lab7-1.asm
```

Рис. 4.1: рис. 1

2. В этот файл вставляем текст из файла in_out.asm для вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax.(рис. 4.2)

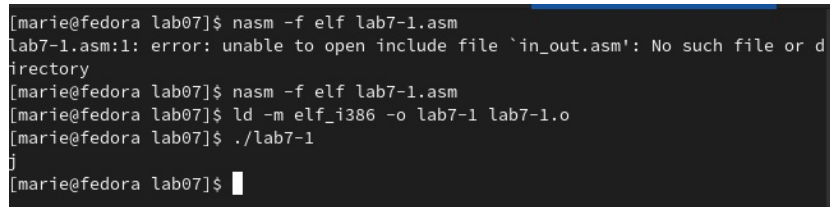


```
Обзор gedit
*lab7-1.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc...

1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1: RESB 80
5
6 SECTION .text
7 GLOBAL _start
8
9 _start:
10
11 mov eax, '6'
12 mov ebx, '4'
13 add eax, ebx
14 mov [buf1], eax
15 mov eax, buf1
16 call sprintLF
17
18 call quit
```

Рис. 4.2: рис. 2

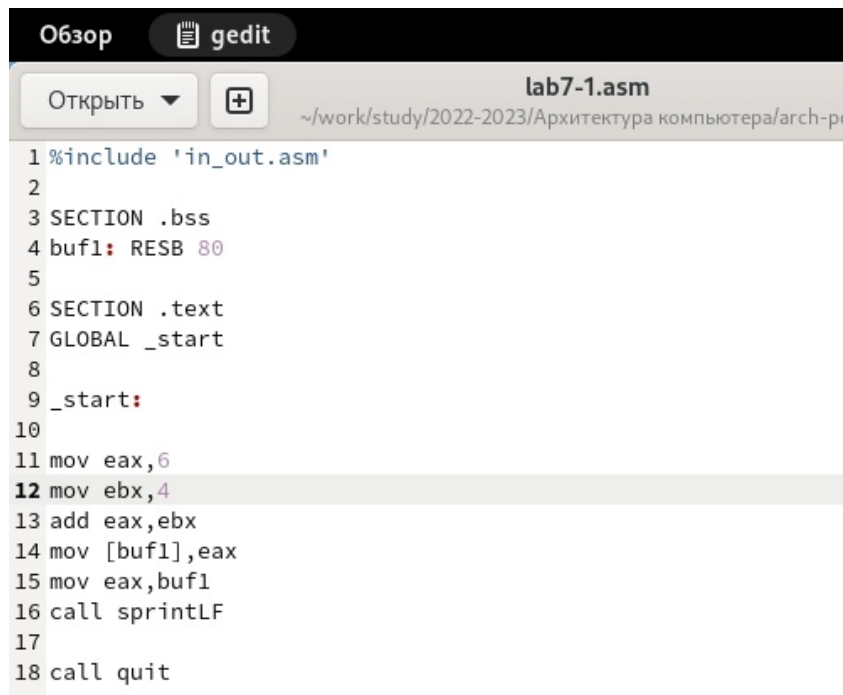
3. Запускаем файл с помощью следующих команд (рис. 4.3) выводим значение. Мы получаем значение j.



```
[marie@fedora lab07]$ nasm -f elf lab7-1.asm
lab7-1.asm:1: error: unable to open include file 'in_out.asm': No such file or d
irectory
[marie@fedora lab07]$ nasm -f elf lab7-1.asm
[marie@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[marie@fedora lab07]$ ./lab7-1
j
[marie@fedora lab07]$
```

Рис. 4.3: рис. 3

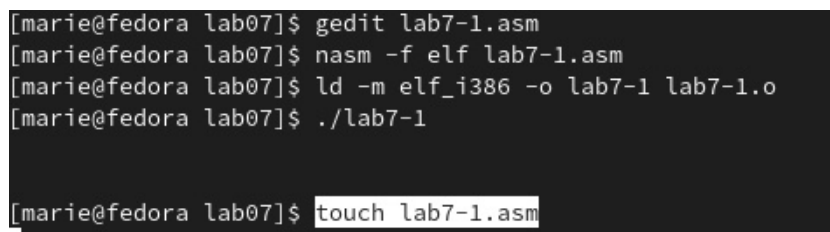
4. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправляем текст программы из рисунка 2 следующим образом: замените строкит `mov eax, '6'`, `mov ebx, '4'` на строки `mov eax, 6`, `mov ebx, 4` (рис. 4.4)



```
Обзор gedit
lab7-1.asm
~/work/study/2022-2023/Архитектура компьютера/arch-p
1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1: RESB 80
5
6 SECTION .text
7 GLOBAL _start
8
9 _start:
10
11 mov eax,6
12 mov ebx,4
13 add eax,ebx
14 mov [buf1],eax
15 mov eax,buf1
16 call sprintf
17
18 call quit
```

Рис. 4.4: рис. 4

5. Создаем файл и запускаем его с помощью команд. Выводится символ с кодом 10, который не отображается на экране. (рис. 4.5)




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

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

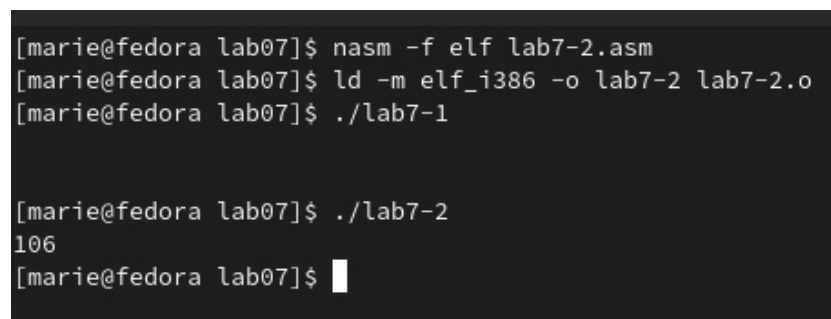
Рис. 4.5: рис. 5

6. Создаем новый файл lab7-2.asm и вставляем исправленный текст из рисунка 2. Создаем файл и запускаем его. В данном случае выводится цифра 106, так как функция iprintfLF позволяет вывести число, а не символ, кодом которого является это число.(рис. 4.6 - 4.7)



```
Открыть ▾ + *lab7-2.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc...
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 call iprintLF
11
12 call quit
```

Рис. 4.6: рис. 6

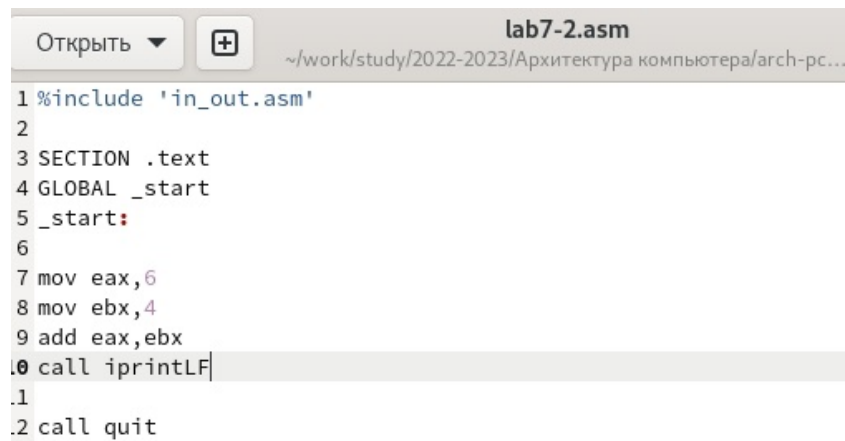


```
[marie@fedora lab07]$ nasm -f elf lab7-2.asm
[marie@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[marie@fedora lab07]$ ./lab7-1

[marie@fedora lab07]$ ./lab7-2
106
[marie@fedora lab07]$
```

Рис. 4.7: рис. 7

7. Аналогично предыдущему примеру изменяем символы на числа. Заменяем строки `mov eax, '6'`, `mov ebx, '4'` на строки `mov eax, 6`, `mov ebx, 4`. Создаем исполняемый файл и запускаем его. (рис. 4.8)



```

1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 call iprintLF
11
12 call quit

```

Рис. 4.8: рис. 8

8. В данном случае выводится цифра 10 (рис. 4.9)




```

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

```

Рис. 4.9: рис. 9

9. Заменяем функцию iprintLF на iprint. Создаем исполняемый файл и запускаем его. (рис. 4.10 - 4.11)



```

1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 call iprint
11
12 call quit

```

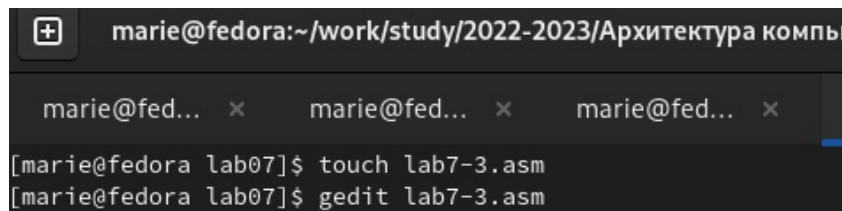
Рис. 4.10: рис. 10

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

Рис. 4.11: рис. 11

4.2 Выполнение арифметических операций в NASM

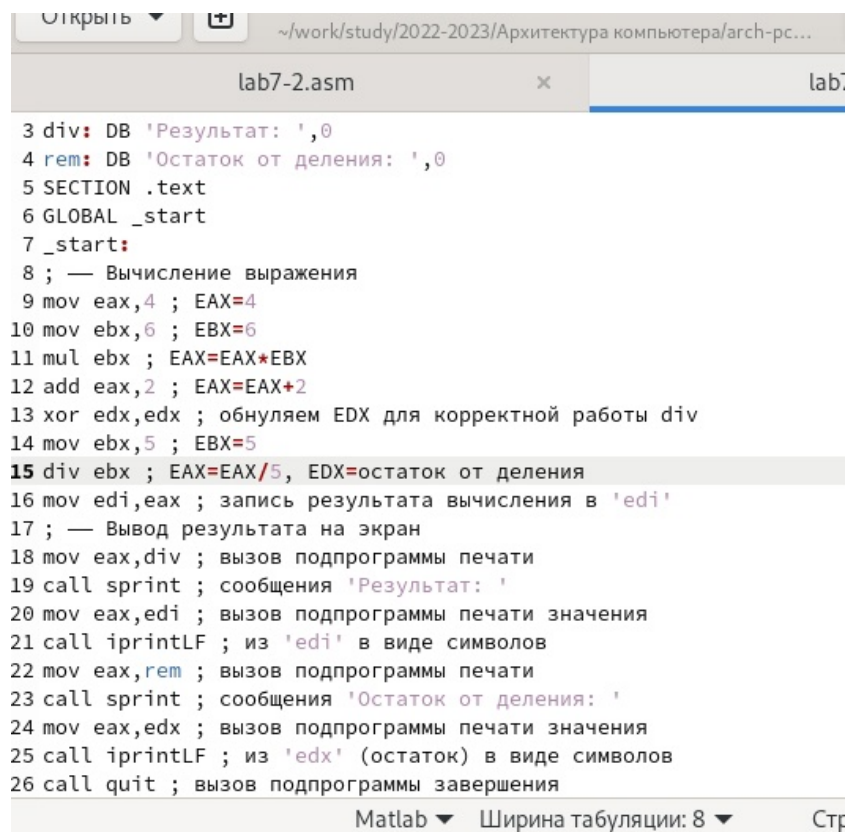
1. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $\boxed{x}(\boxed{x}) = (5 \boxed{x} 2 + 3)/3$. Создаем файл lab7-3.asm в каталоге ~/work/arch-рс/lab07 и открываем его с помощью редактора (рис. 4.12)



```
marie@fedora:~/work/study/2022-2023/Архитектура компь
marie@fed... x marie@fed... x marie@fed... x
[marie@fedora lab07]$ touch lab7-3.asm
[marie@fedora lab07]$ gedit lab7-3.asm
```

Рис. 4.12: рис. 12

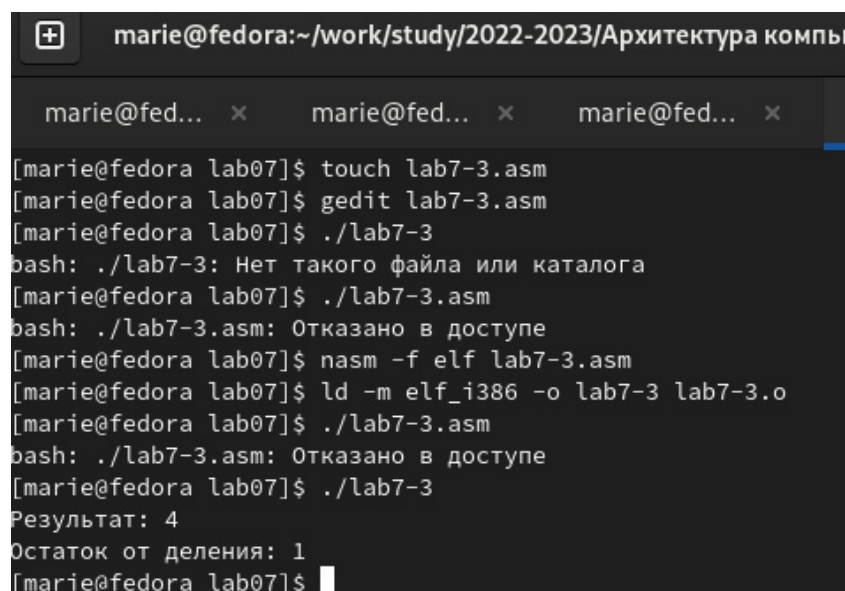
2. В этот файл вставляем следующий текст (рис. 4.13)



```
Открыть ~/work/study/2022-2023/Архитектура компьютера/arch-pc...
lab7-2.asm x lab7-2.asm
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; — Вычисление выражения
9 mov eax,4 ; EAX=4
10 mov ebx,6 ; EBX=6
11 mul ebx ; EAX=EAX*EBX
12 add eax,2 ; EAX=EAX+2
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,5 ; EBX=5
15 div ebx ; EAX=EAX/5, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; — Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения
Matlab Ширина табуляции: 8 Стр
```

Рис. 4.13: рис. 13

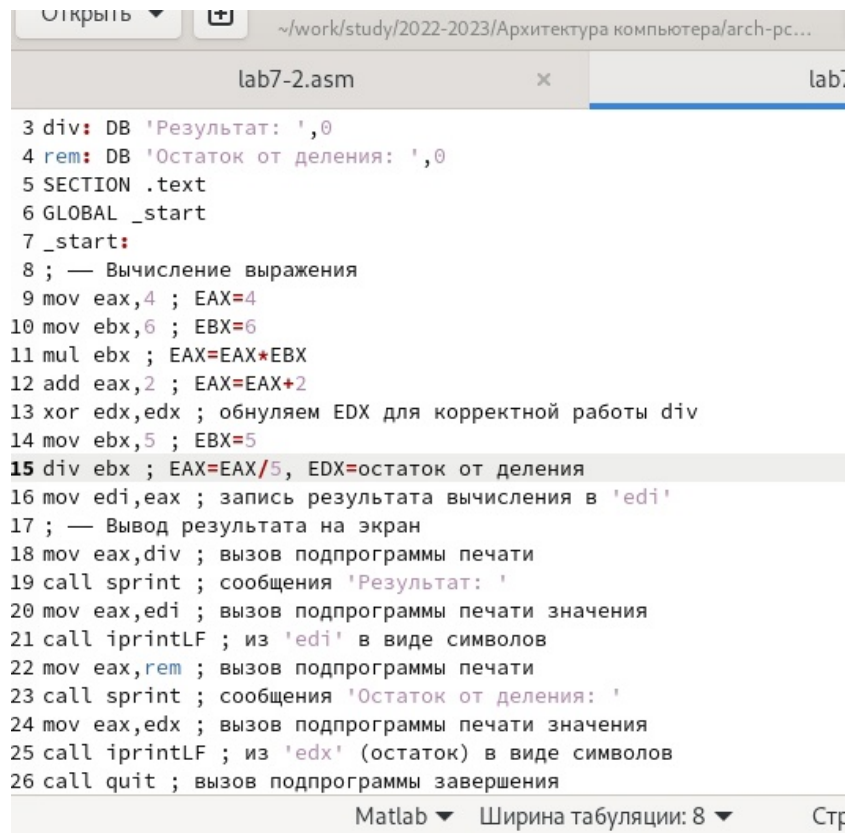
3. Создаем файл и запускаем его. Результат работы следующий: (рис. 4.14)



```
marie@fedora:~/work/study/2022-2023/Архитектура компь
marie@fed... x marie@fed... x marie@fed... x
[marie@fedora lab07]$ touch lab7-3.asm
[marie@fedora lab07]$ gedit lab7-3.asm
[marie@fedora lab07]$ ./lab7-3
bash: ./lab7-3: Нет такого файла или каталога
[marie@fedora lab07]$ ./lab7-3.asm
bash: ./lab7-3.asm: Отказано в доступе
[marie@fedora lab07]$ nasm -f elf lab7-3.asm
[marie@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[marie@fedora lab07]$ ./lab7-3.asm
bash: ./lab7-3.asm: Отказано в доступе
[marie@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[marie@fedora lab07]$
```

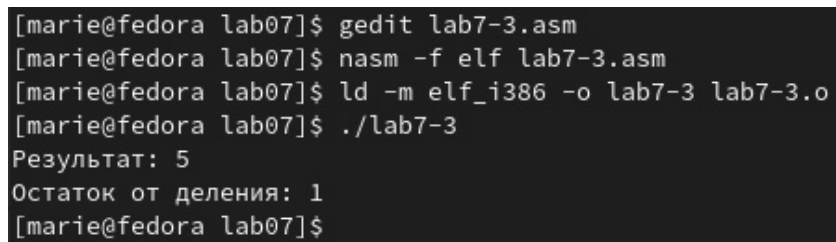
Рис. 4.14: рис. 14

4. Изменяем текст программы для вычисления выражения $\boxed{x}(\boxed{x}) = (4 \boxed{x} 6 + 2)/5$.
Создаем исполняемый файл и проверяем его работу. (рис. 4.15 - 4.16)



```
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; — Вычисление выражения
9 mov eax,4 ; EAX=4
10 mov ebx,6 ; EBX=6
11 mul ebx ; EAX=EAX*EBX
12 add eax,2 ; EAX=EAX+2
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,5 ; EBX=5
15 div ebx ; EAX=EAX/5, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; — Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения
```

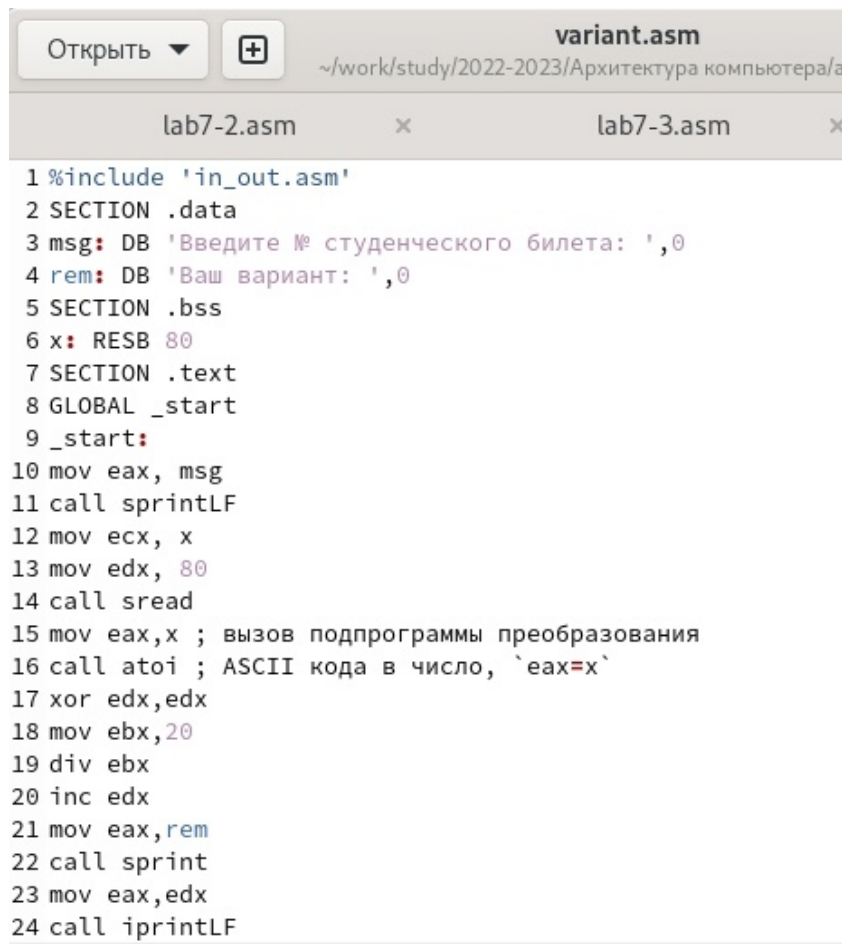
Рис. 4.15: рис. 15



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

Рис. 4.16: рис. 16

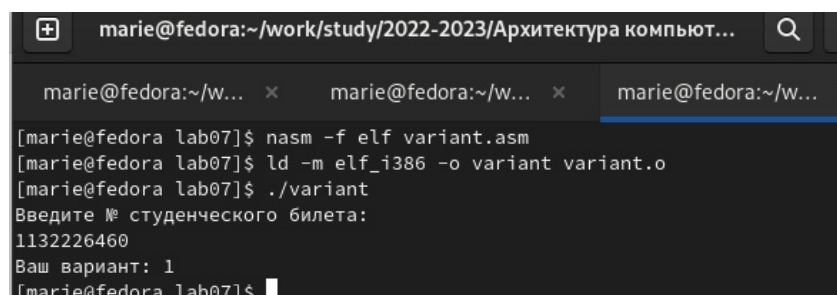
5. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. Создаем файл variant.asm в каталоге ~/work/arch-pc/lab07 с помощью команды touch. В этот файл вводим следующий текст (рис. 4.17)



```
variant.asm
~/work/study/2022-2023/Архитектура компьютера/a
lab7-2.asm x lab7-3.asm x
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите № студенческого билета: ',0
4 rem: DB 'Ваш вариант: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintLF
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax,x ; вызов подпрограммы преобразования
16 call atoi ; ASCII кода в число, `eax=x`
17 xor edx,edx
18 mov ebx,20
19 div ebx
20 inc edx
21 mov eax,rem
22 call sprint
23 mov eax,edx
24 call iprintLF
```

Рис. 4.17: рис. 17

6. Проверяем работу, ввожу номер студенческого билета, он выводит номер варианта(рис. 4.18).



```
marie@fedora:~/work/study/2022-2023/Архитектура компьют...
marie@fedora:~/w... x marie@fedora:~/w... x marie@fedora:~/w...
[marie@fedora lab07]$ nasm -f elf variant.asm
[marie@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[marie@fedora lab07]$ ./variant
Введите № студенческого билета:
1132226460
Ваш вариант: 1
[marie@fedora lab07]$
```

Рис. 4.18: рис. 18

4.3 Ответы на вопросы по программе

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`

4.4 Самостоятельная работа

Вариант 1, вычисление $(10 + 2 \times \square)/3$

1. создаем файл `lab4-1.asm`, открываем его с помощью редактора `gedit` и вставляем следующий текст (рис. 4.19 - 4.20)

```
[marie@fedora lab07]$ touch lab7-4.asm
[marie@fedora lab07]$ gedit lab7-4.asm
[marie@fedora lab07]$ nasm -f elf lab7-4.asm
[marie@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
```

Рис. 4.19: рис. 19

```
msg: db "Введите значение переменной x: ",0
rem: db "Результат: ",0
SECTION .bss ; секция не иницированных данных
x: resb 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; — Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в eax
call sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
mul ebx, 2; EAX=EAX*EBX =2*x
add eax, 10; eax = 2* eax+10 = 2*x + 10
mov ebx, 3 ; запись значения 3 в регистр ebx
div ebx; EAX=EAX/EBX = (2*x+11)/3
mov edi, eax ; запись результата вычисления в `edi`
; — Вывод результата на экран
mov eax, rem ; вызов подпрограммы печати
call sprint ; сообщения "Результат: "
mov eax, edi ; вызов подпрограммы печати значения
call iprint ; из `edi` в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.20: рис. 20

2. Создаем и запускаем файл. Получаем следующие результаты: (рис. 4.21 - 4.22)

```
[marie@fedora lab07]$ ./lab7-4
Введите значение переменной x: 1
Результат: 18[marie@fedora lab07]$
```

Рис. 4.21: рис. 21

```
[marie@fedora lab07]$ ./lab7-4
Введите значение переменной x: 10
Результат: 36[marie@fedora lab07]$
```

Рис. 4.22: рис. 22

5 Выводы

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

Список литературы

1. Лабораторная работа №7
2. Таблица ASCII