

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Ермакова Анастасия Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Символьные и численные данные в NASM	9
4.2	Выполнение арифметических операций в NASM	12
4.2.1	Ответы на вопросы	15
4.3	Выполнение заданий для самостоятельной работы	16
5	Выводы	20
6	Список литературы	21

Список иллюстраций

4.1	Создание каталога для лабораторной работы	9
4.2	Создание файла lab6-1.asm	9
4.3	Создание исполняемого файла. Результат его работы	9
4.4	Измененный текст программы	10
4.5	Результат работы исполняемого файла	10
4.6	Создание файла lab6-2.asm	10
4.7	Текст программы	11
4.8	Результат работы исполняемого файла	11
4.9	Измененный текст программы	11
4.10	Результат работы программы	12
4.11	Результат работы измененного файла	12
4.12	Текст программы файла lab6-3.asm	13
4.13	Результат работы программы	13
4.14	Измененный текст программы	14
4.15	Результат работы измененной программы	14
4.16	Текст программы файла variant.asm	15
4.17	Результат работы программы	15
4.18	Создание файла lab6-4.asm	16
4.19	Текст программы файла lab6-4.asm	17
4.20	Результат работы программы	18

Список таблиц

1 Цель работы

Цель данной лабораторной работы - освоить арифметические инструкции 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.

- 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. Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Расширенная таблица ASCII состоит из двух частей. Первая (символы с кодами 0-127) является универсальной (см. Приложение.), а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят

как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

Создаю каталог для программ лабораторной работы №6. (рис. 4.1)

```
anastasia@fedora: ~$ mkdir ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab06
```

Рис. 4.1: Создание каталога для лабораторной работы

Перехожу в созданный каталог и создаю файл lab6-1.asm. (рис. 4.2)

```
anastasia@fedora: ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab06$ touch lab6-1.asm
```

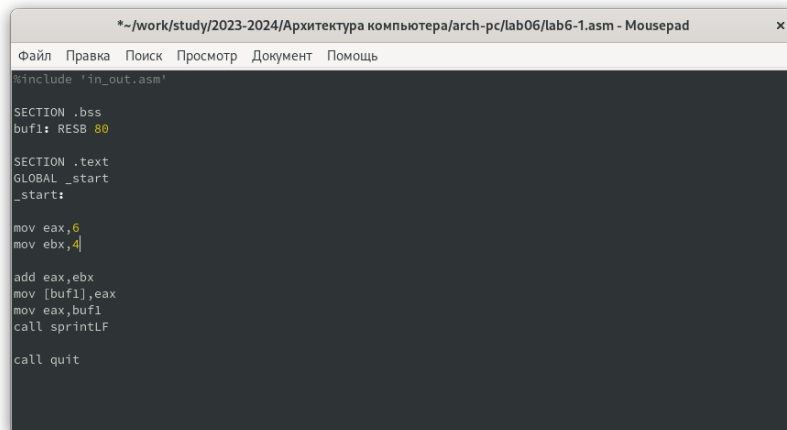
Рис. 4.2: Создание файла lab6-1.asm

Открываю файл с помощью текстового редактора mousepad и ввожу текст программы из листинка 6.1. Создаю исполняемый файл и запускаю его (рис. 4.3) Программа вывела символ j, т.к. он соответствует сумме двоичных кодов символов 4 и 6 по системе ASCII.

```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ mousepad lab6-1.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-1.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ./lab6-1
j
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.3: Создание исполняемого файла. Результат его работы

Вношу изменения в текст программы, вместо символов записываю регистры числа, заменяю 'б' и '4' на б и 4. (рис. 4.4)



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06/lab6-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include 'in_out.asm'

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

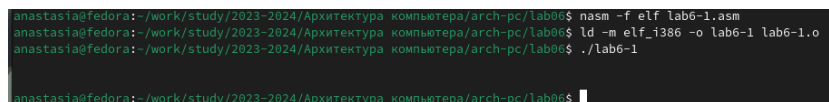
mov eax,6
mov ebx,4

add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintLF

call quit
```

Рис. 4.4: Измененный текст программы

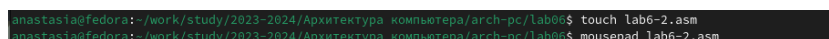
Создаю исполняемый файл и запускаю его (рис. 4.5) Результатом работы является символ с кодом 10, то есть символ перевода строки. На экран этот символ не отображается.



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-1.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ./lab6-1
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.5: Результат работы исполняемого файла

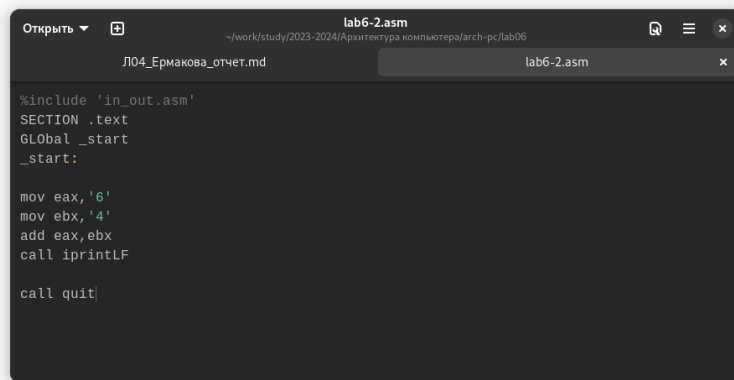
Создаю файл lab6-2.asm и открывю его с помощью текстового редактора. (рис. 4.6)



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ touch lab6-2.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ mousepad lab6-2.asm
```

Рис. 4.6: Создание файла lab6-2.asm

Ввожу в него текст программы из листинга 6.2. (рис. 4.7)



```
lab6-2.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06
ЛО4_Ермакова_отчет.md lab6-2.asm

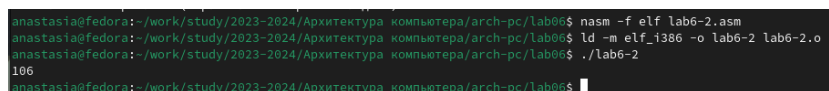
%include 'in_out.asm'
SECTION .text
Global _start
_start:

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

call quit
```

Рис. 4.7: Текст программы

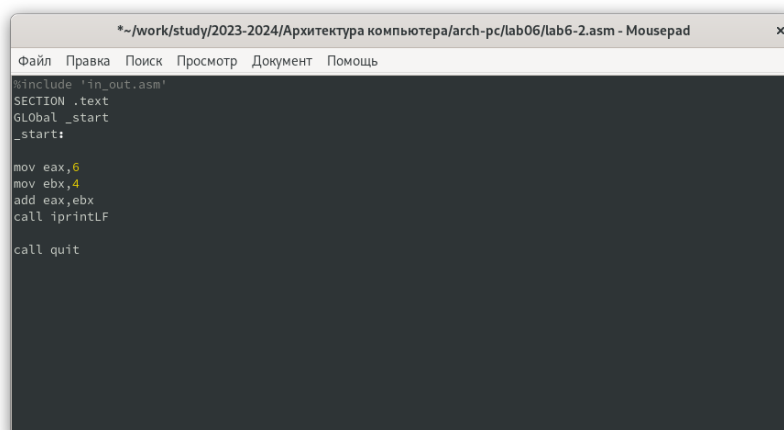
Создаю исполняемый файл и запускаю его (рис. 4.8) Программа вывела число 106, потому что она позволяет вывести именно число, а не код, после сложения символов '6' и '4'.



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-2.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ./lab6-2
106
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.8: Результат работы исполняемого файла

Изменяю текст программы, заменяю '6' и '4' на числа 6 и 4. (рис. 4.9)



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06/lab6-2.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь

%include 'in_out.asm'
SECTION .text
Global _start
_start:

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

call quit
```

Рис. 4.9: Измененный текст программы

Создаю исполняемый файл и запускаю его (рис. 4.10) Программа сложила 6 и 4, поэтому выводит число 10.

```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-2.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ./lab6-2
10
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.10: Результат работы программы

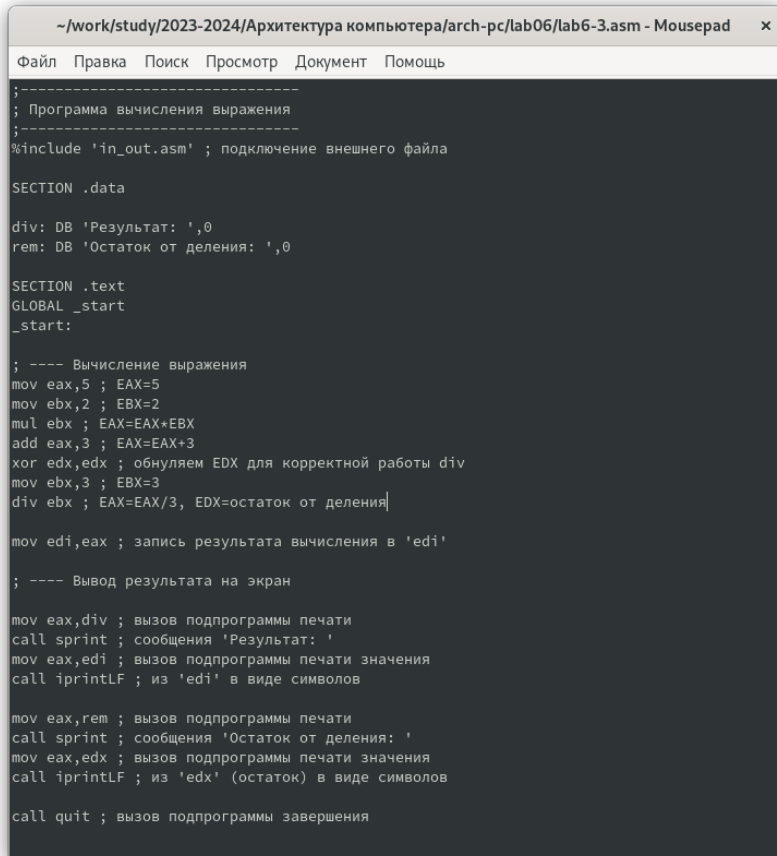
Заменяю в тексте программы функцию `iprintLF` на `iprint`. (рис. 4.11) Функция `iprint` не добавляет к выводу символ переноса строки, в отличие от функции `iprintLF`, поэтому после числа 10 нет переноса.

```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ mousepad lab6-2.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-2.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ./lab6-2
10anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.11: Результат работы измененного файла

4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью утилиты `touch` и ввожу в него текст программы из листинга 6.3 (рис. 4.12)



```
~\work\study\2023-2024\Архитектура компьютера\arch-pc\lab06\lab6-3.asm - Mousepad x
Файл  Правка  Поиск  Просмотр  Документ  Помощь

;-----
; 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

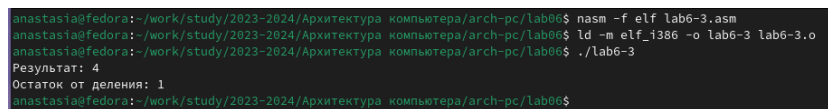
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов

call quit ; вызов подпрограммы завершения
```

Рис. 4.12: Текст программы файла lab6-3.asm

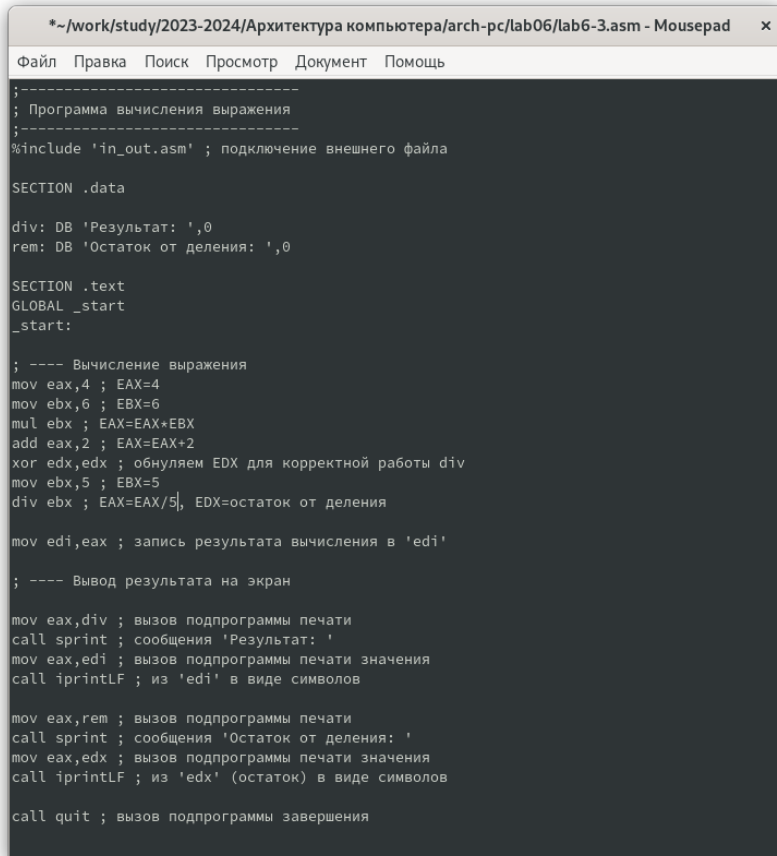
Создаю исполняемый файл и запускаю его (рис. 4.13) Результат работы программы совпадает с тем, что представлен в файле на ТУИС.



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-3.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.13: Результат работы программы

Вношу изменения в текст программы, чтобы она посчитала значение выражения $(4*6 + 2)/5$ (рис. 4.14)



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06/lab6-3.asm - Mousepad x
Файл  Правка  Поиск  Просмотр  Документ  Помощь
;-----
; 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,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'

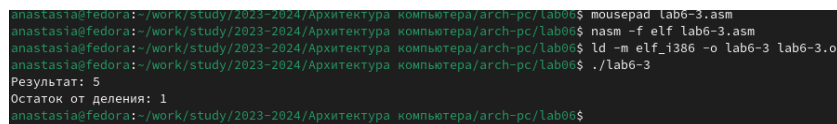
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов

call quit ; вызов подпрограммы завершения
```

Рис. 4.14: Измененный текст программы

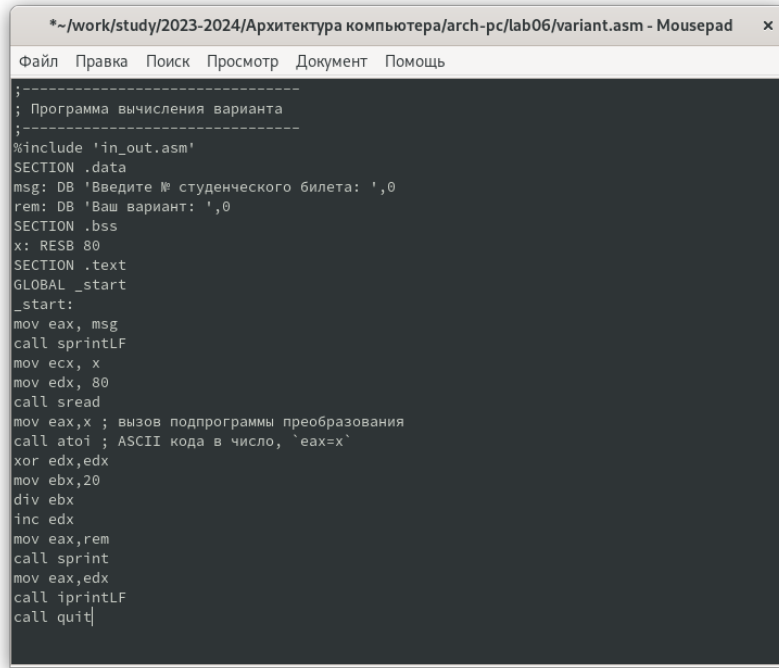
Создаю исполняемый файл и запускаю его (рис. 4.15) Результатом является 5, а остатком от деления 1. Программа работает верно.



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ mousepad lab6-3.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-3.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.15: Результат работы измененной программы

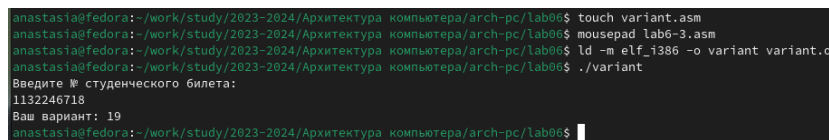
Далее создаю файл variant.asm и ввожу в него текст программы из листинга 6.4 (рис. 4.16)



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06/variant.asm - Mousepad x
Файл  Правка  Поиск  Просмотр  Документ  Помощь
;-----
; 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
mov ebx,20
div ebx
inc edx
mov eax,rem
call sprint
mov eax,edx
call iprintf
call quit
```

Рис. 4.16: Текст программы файла variant.asm

Создаю исполняемый файл и запускаю его (рис. 4.17) Программа вычислила вариант задания по номеру студенческого билета, который я ввела с клавиатуры. На экран вывелся мой вариант (19).



```
anastasiya@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ touch variant.asm
anastasiya@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ mousepad lab6-3.asm
anastasiya@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
anastasiya@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246718
Ваш вариант: 19
anastasiya@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.17: Результат работы программы

4.2.1 Ответы на вопросы

1. За вывод сообщения “Ваш вариант” отвечают следующие строки кода:

```
mov eax,rem
call sprint
```

2. Инструкция `mov ecx, x` используется для того, чтобы положить адрес вводимой строки `x` в регистр `ecx`. Инструкция `mov edx,80` используется для

того, чтобы записать в регистр `edx` длины вводимой строки. Инструкция `call streadd` - для вызова подпрограммы из внешнего файла для ввода сообщения с клавиатуры.

3. Инструкция `call atoi` используется для вызова подпрограммы из внешнего файла для преобразования ASCII-кода символа в целое число и записывает результат в регистр `eax`.

4. За вычисление варианта отвечают следующие 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`.

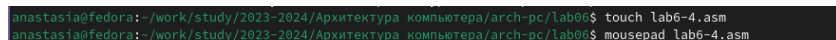
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1.

7. За вывод на экран результатов вычислений отвечают следующие строки:

```
mov eax,edx
call sprintf
```

4.3 Выполнение заданий для самостоятельной работы

Создаю файл `lab6-4.asm` с помощью утилиты `touch` и открываю файл для редактирования. (рис. 4.18)

A screenshot of a terminal window with a dark background. It shows two lines of text: the first line shows the command 'touch lab6-4.asm' being executed, and the second line shows the command 'mousepad lab6-4.asm' being executed, opening the file in a text editor.

```
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ touch lab6-4.asm
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ mousepad lab6-4.asm
```

Рис. 4.18: Создание файла `lab6-4.asm`

Ввожу в созданный файл текст программы для вычисления выражения под номером 19: $(1/3x + 5)7$ (рис. 4.19)

```
%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
call atoi

xor edx, edx
mov ebx, 3
div ebx
add eax, 5
mov ebx, 7
mul ebx

mov edi, eax

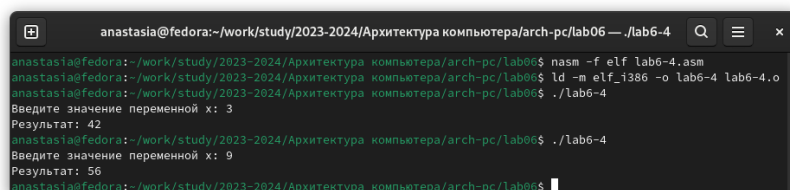
mov eax, rem
call sprint
mov eax, edi
call iprintLF

call quit
```

Рис. 4.19: Текст программы файла lab6-4.asm

Создаю исполняемый файл и запускаю его (рис. 4.20) Ввожу с клавиатуры значение переменной 3, результат выводится на экран - ответ 42. Повторно запускаю

программу и ввожу 9, результатом является 56. Программа работает исправно, ответы верные.



```
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 — ./lab6-4
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-4.asm
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 3
Результат: 42
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 9
Результат: 56
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.20: Результат работы программы

Код программы для вычисления значения выражения $(1/3x + 5)7$:

```
%include 'in_out.asm'          ; подключение внешнего файла

SECTION .data                  ; секция инициированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0

SECTION .bss                   ; секция не инициированных данных
x: RESB 80                     ; переменная, значение которой будем вводить с клавиатуры

SECTION .text                  ; код программы
GLOBAL _start                 ; начало программы
_start:                       ; точка входа в программу

mov eax, msg                   ; запись адреса вводимого сообщения в eax
call sprint                    ; вызов подпрограммы печати сообщения

mov ecx, x                     ; запись адреса переменной x в ecx
mov edx, 80                    ; запись длины вводимого сообщения в edx
call sread                     ; вызов подпрограммы ввода сообщения
```

```

mov eax, x           ; вызов подпрограммы преобразования
call atoi            ; ASCII кода в число 'eax=x'

xor edx,edx           ; обнуление edx для корректной работы div
mov ebx,3             ; ebx = 3
div ebx              ; eax = eax/3, edx - остаток от деления
add eax,5             ; eax = eax+5
mov ebx,7             ; ebx = 7
mul ebx              ; eax = eax*ebx

mov edi,eax           ; запись результата вычисления в 'edi'

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

call quit             ; вызов подпрограммы завершения

```

5 Выводы

В ходе выполнения лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

6 Список литературы

1. Архитектура ЭВМ