

Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютера

Хатамов Эзиз

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
4.2.1	В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $\square(\square) = (5 \square 2 + 3)/3$	13
4.2.2	В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:	14
4.3	Ответы на вопросы по программе	15
5	Выполнение заданий для самостоятельной работы	17
6	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Создания каталога и файла	8
4.2	Копирования файла на нужный каталог	9
4.3	Изменения в тексте файла	9
4.4	Изменения '6' и '4' на 6 и 4	9
4.5	Создания нового файла lab6-2.asm	10
4.6	Изменения текста файла lab6-2.asm	10
4.7	Создание и запуск исполняемого файла	10
4.8	Изменения eax и ebx	11
4.9	Создание и запуск исполняемого файла(измененного eax)	11
4.10	Изменения iprintLF на iprint	12
4.11	Создание и запуск исполняемого файла(измененного iprintLF)	12
4.12	Создание файла и изменения текста lab6-3.asm	13
4.13	Создание исполняемого файла lab6-3.asm и запуск файла	13
4.14	Изменения текста lab6-3.asm	14
4.15	Создание исполняемого файла lab6-3.asm и запуск файла (с изменением)	14
4.16	Создание файла и изменения текста variant.asm	15
4.17	Создание исполняемого файла lab6-3.asm и запуск файла	15
5.1	Создание файла и изменения текста lab6-4.asm	17
5.2	Создание исполняемого файла lab6-4.asm и запуск файла	18
5.3	Запуск исполняемго файла lab6-4.asm	18

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

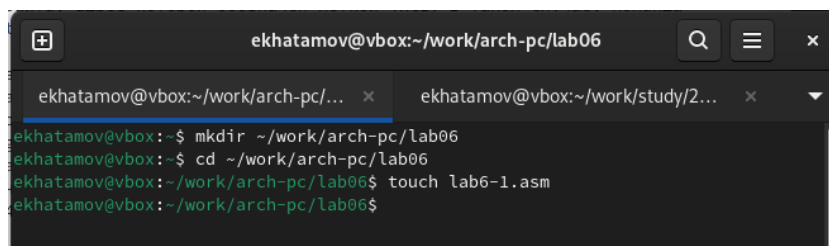
3 Теоретическое введение

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. Например, определим переменную `intg DD 3` – это означает, что задается область памяти размером 4 байта, адрес которой обозначен меткой `intg`. В таком случае, команда `mov eax,[intg]` копирует из памяти по адресу `intg` данные в регистр `eax`. В свою очередь команда `mov [intg],eax` запишет в память по адресу `intg` данные из регистра `eax`. Также рассмотрим команду `mov eax,intg` В этом случае в регистр `eax` запишется адрес `intg`. Допустим, для `intg` выделена память начиная с ячейки с адресом `0x600144`, тогда команда `mov eax,intg` аналогична команде `mov eax,0x600144` – т.е. эта команда запишет в регистр `eax` число `0x600144`.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

Для начала я создал каталог для программ лабораторной работы №6, потом перешел на этот каталог и создал файл lab6-1.asm (рис. 4.1).



```
ekhatamov@vbox:~/work/arch-pc/lab06
ekhatamov@vbox:~/work/arch-pc/... x ekhatamov@vbox:~/work/study/2... x
ekhatamov@vbox:~$ mkdir ~/work/arch-pc/lab06
ekhatamov@vbox:~$ cd ~/work/arch-pc/lab06
ekhatamov@vbox:~/work/arch-pc/lab06$ touch lab6-1.asm
ekhatamov@vbox:~/work/arch-pc/lab06$
```

Рис. 4.1: Создания каталога и файла

Потом я зашел на Midnight Commander и скопировал in_out.asm в каталог с файлом lab6-1.asm с помощью клавиши F5

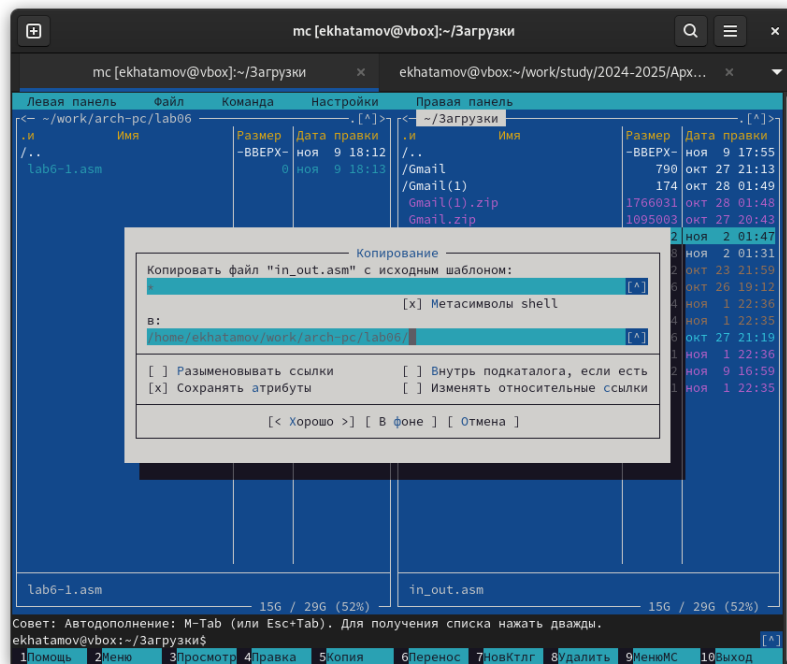


Рис. 4.2: Копирования файла на нужный каталог

Потом я открыл созданный файл lab6-1.asm и внес изменения в тексте файла.

```
ekhatamov@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
ekhatamov@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
ekhatamov@vbox:~/work/arch-pc/lab06$ ./lab6-1
j
ekhatamov@vbox:~/work/arch-pc/lab06$
```

Рис. 4.3: Изменения в тексте файла

потом в тексте я изменил еах,'6' на еах,6 а еbx,'4' на еbx,4 .

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

ekhatamov@vbox:~/work/arch-pc/lab06$
```

Рис. 4.4: Изменения '6' и '4' на 6 и 4

Все равно не вышло число 10 а вместо него вышло пустота. я зашел и посмотрел таблицу ASCII и там я увидел что символ числа 10 это пустота

Потом я создал новый файл в том же каталоге в котором был прошлый файл

```
ekhatamov@vbox:~/work/arch-pc/lab06$ touch lab6-2.asm
ekhatamov@vbox:~/work/arch-pc/lab06$
```

Рис. 4.5: Создания нового файла lab6-2.asm

После того как я создал файл я зашел на него и изменил текст файла

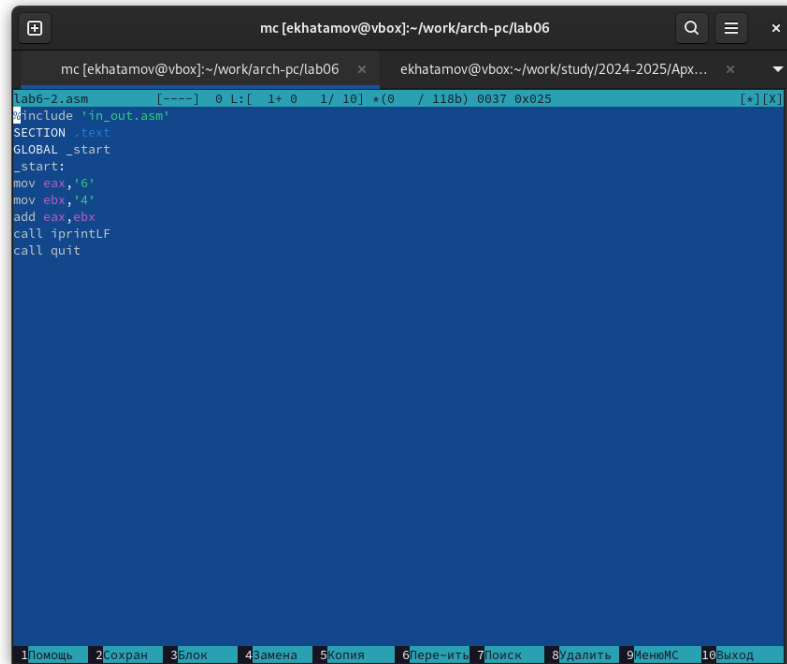


Рис. 4.6: Изменения текста файла lab6-2.asm

Создал исполняемый файл и запустил его

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

Рис. 4.7: Создание и запуск исполняемого файла

Потом заменил где еах '6' и еах '4' на ебх,6 и ебх,4

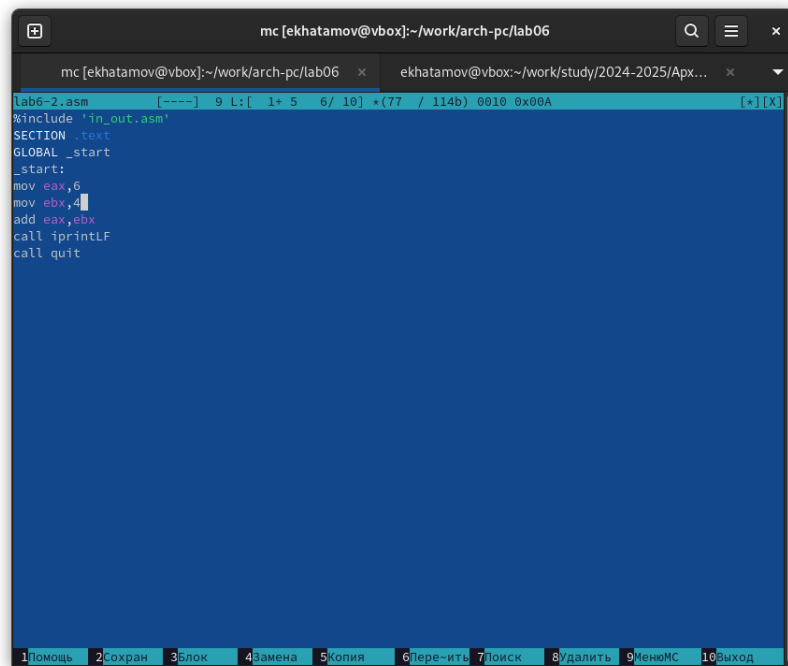


Рис. 4.8: Изменения eax и ebx

После изменения я создал исполняемый файл и запустил его

```

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

```

Рис. 4.9: Создание и запуск исполняемого файла(измененного eax)

После этих изменений я получил цифру 10. Потом я изменил iprintLF на iprint

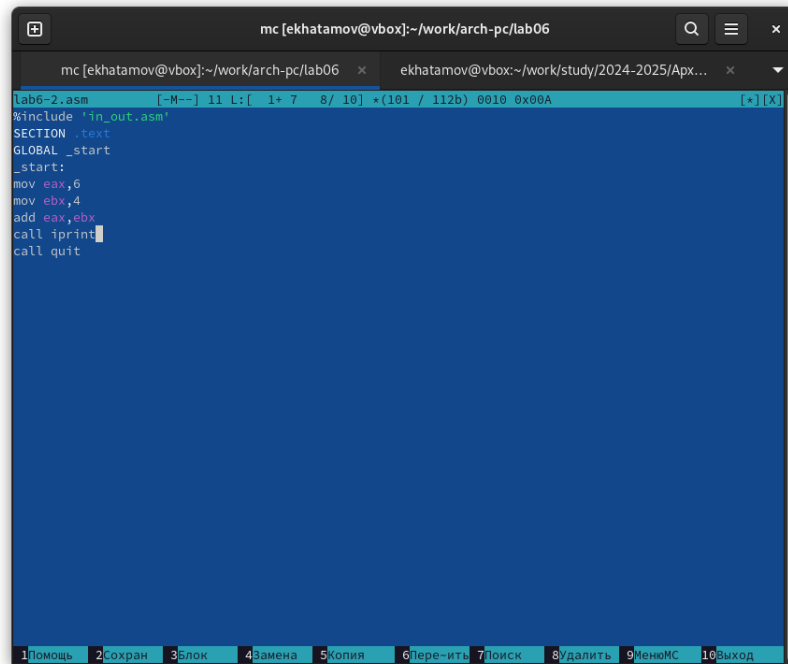


Рис. 4.10: Изменения iprintLF на iprint

После изменения я занова создал исполняемый файл и запустил его

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

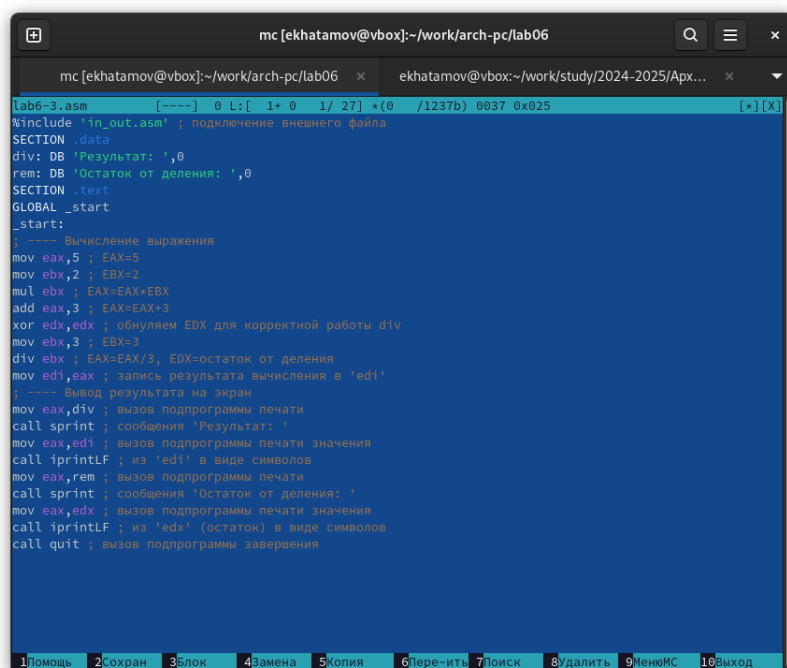
Рис. 4.11: Создание и запуск исполняемого файла(измененного iprintLF)

Таким образом, разница между `iprintLF` и `iprint` в NASM заключается в том, что `iprintLF` — это функция для печати целых чисел с переводом на новую строку, а `iprint` — для простой печати целых чисел без перевода на новую строку

4.2 Выполнение арифметических операций в NASM

4.2.1 В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $\lfloor(5 \div 2 + 3) / 3\rfloor$.

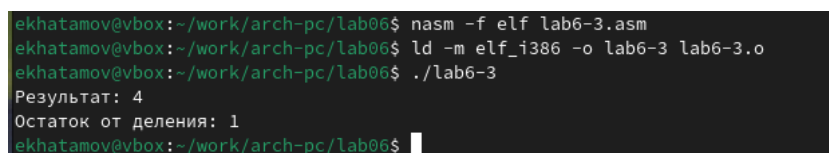
Для начала я создал файл lab6-3.asm с помощью touch потом внес изменения в текст файла



```
lab6-3.asm [-----] 0 L: [ 1+ 0 1/ 27] * (0 /1237b) 0037 0x025 [*] [X]
%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 iprintlnf ; 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 iprintlnf ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.12: Создание файла и изменения текста lab6-3.asm

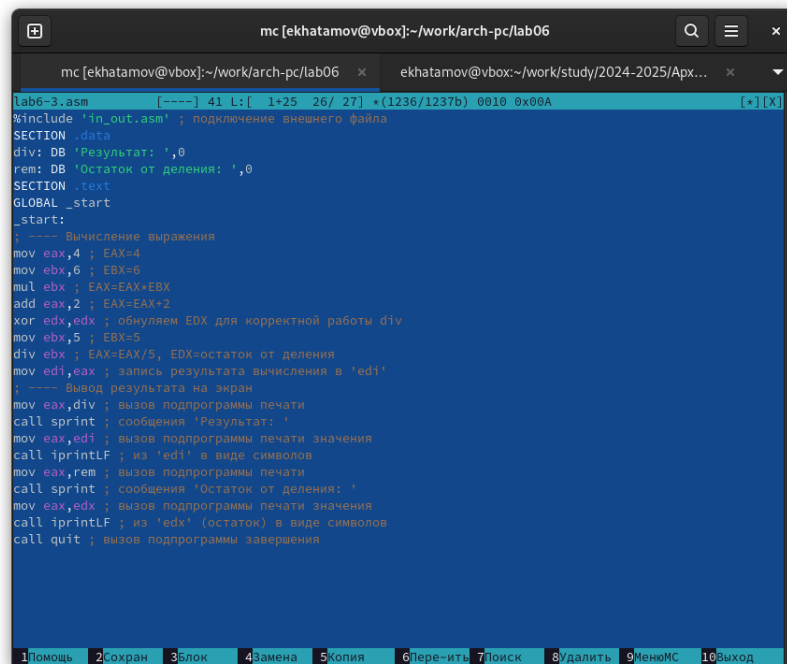
Потом создал исполняемый файл lab6-3.asm и запустил его



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

Рис. 4.13: Создание исполняемого файла lab6-3.asm и запуск файла

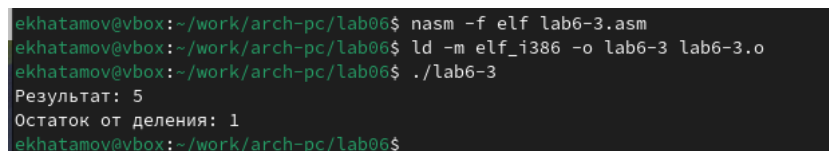
Потом изменил текст программы для вычисления выражения $\lfloor(4 \div 6 + 2) / 5\rfloor$



```
lab6-3.asm [-----] 41 L: [ 1+25 26/ 27] *(1236/1237b) 0010 0x00A [*] [X]
#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 iprintfLF ; 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 iprintfLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.14: Изменения текста lab6-3.asm

Потом создал исполняемый файл lab6-3.asm и запустил его

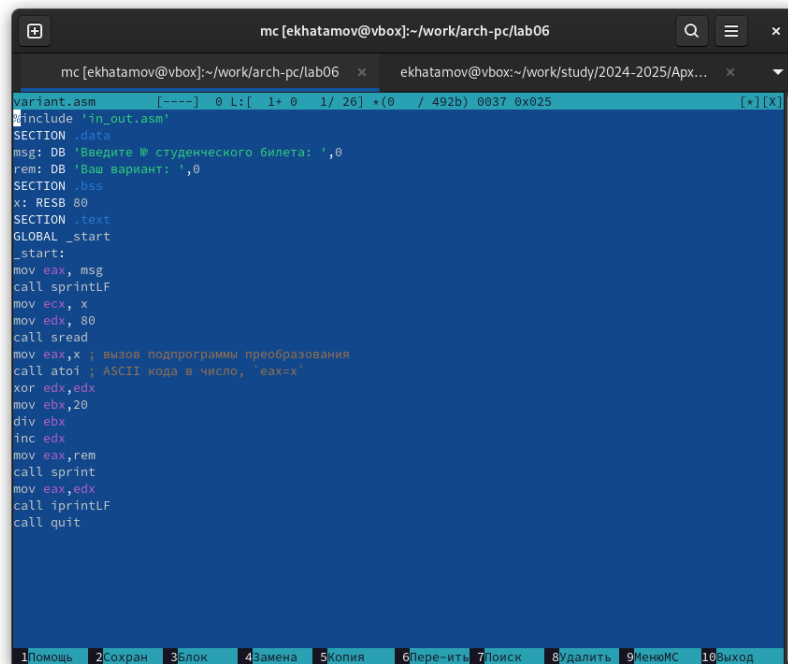


```
ekhatamov@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
ekhatamov@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
ekhatamov@vbox:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
ekhatamov@vbox:~/work/arch-pc/lab06$
```

Рис. 4.15: Создание исполняемого файла lab6-3.asm и запуск файла (с изменением)

4.2.2 В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

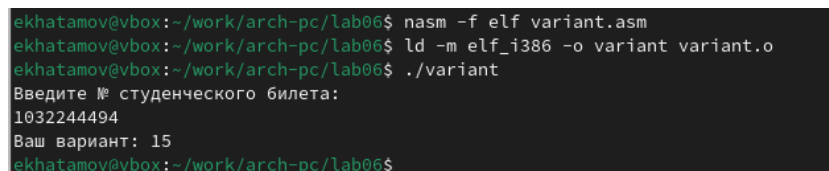
Для начала я создал файл variant.asm с помощью команды touch и внес в него изменения



```
variant.asm [----] 0 L: [ 1+ 0 1/ 26] * (0 / 492b) 0037 0x025 [*] [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 ; 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 sprintf
mov eax, edx
call iprintf
call quit
```

Рис. 4.16: Создание файла и изменения текста variant.asm

Потом создал исполняемый файл запустил его



```
ekhatamov@vbox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
ekhatamov@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
ekhatamov@vbox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
103224494
Ваш вариант: 15
ekhatamov@vbox:~/work/arch-pc/lab06$
```

Рис. 4.17: Создание исполняемого файла lab6-3.asm и запуск файла

4.3 Ответы на вопросы по программе

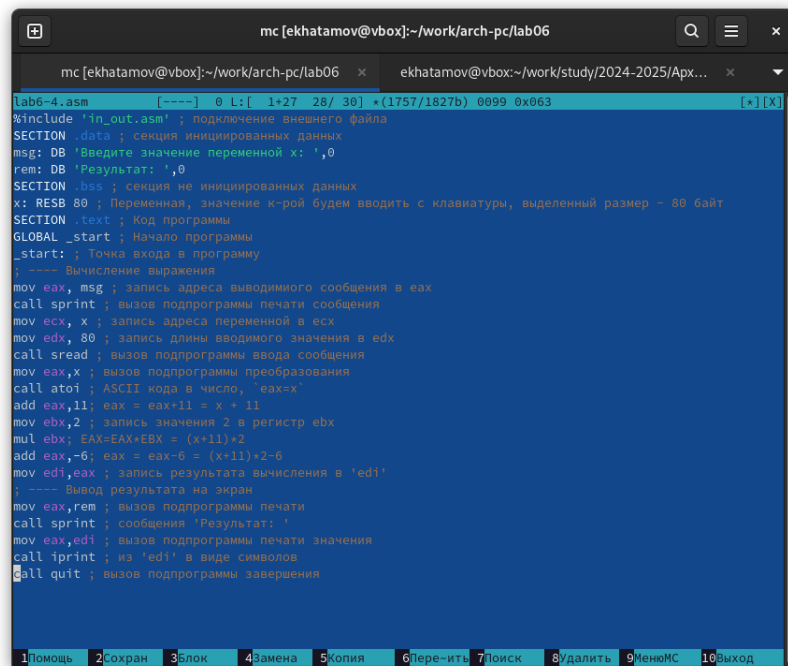
1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант: '? Ответ: За вывод сообщения "Ваш вариант" отвечают строки кода:
`mov eax, rem`
`call sprintf`
2. Для чего используются следующие инструкции? `mov ecx, x` `mov edx, 80` `call sread` Ответ: Инструкция `mov ecx, x` используется, чтобы положить адрес вводимой строки x в регистр ecx `mov edx, 80` - запись в регистр edx для-

ны вводимой строки `call sread` - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры

3. Для чего используется инструкция “`call atoi`”? Ответ: Инструкция «`call atoi`» используется для преобразования строки в целое число.
4. Какие строки листинга 6.4 отвечают за вычисления варианта? Ответ: `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. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”? Ответ: При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`
6. Для чего используется инструкция “`inc edx`”? Ответ: Инструкция `inc edx` увеличивает значение регистра `edx` на 1
7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений? Ответ: За вывод на экран результатов вычислений отвечают строки: `mov eax,edx` `call iprintLF`

5 Выполнение заданий для самостоятельной работы

Для начала я создал файл lab6-4.asm с помощью touch. И открыл файл для редактирования, ввел туда текст программы для вычисления $(11+x)*2-6$



```
mc [ekhatamov@vbox]:~/work/arch-pc/lab06
lab6-4.asm [----] 0 L: [ 1+27 28/ 30] *(1757/1827b) 0099 0x063 [*] [X]
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; секция инициализированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss ; секция не инициализированных 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'
add eax, 11; eax = eax + 11 = x + 11
mov ebx, 2 ; запись значения 2 в регистр ebx
mul ebx; EAX=EAX*EBX = (x+11)*2
add eax, -6; eax = eax - 6 = (x+11)*2 - 6
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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пере-ить 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 5.1: Создание файла и изменения текста lab6-4.asm

Потом создал исполняемый файл и запустил его. И ввел туда цифру 1

```
ekhatamov@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
ekhatamov@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
ekhatamov@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 18
ekhatamov@vbox:~/work/arch-pc/lab06$
```

Рис. 5.2: Создание исполняемого файла lab6-4.asm и запуск файла

Еще раз запустил файл но в этот раз ввел цифру 9 и по алгоритму отработала верно и дал верный ответ

```
ekhatamov@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
ekhatamov@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
ekhatamov@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 18
ekhatamov@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 9
Результат: 34
ekhatamov@vbox:~/work/arch-pc/lab06$
```

Рис. 5.3: Запуск исполняемого файла lab6-4.asm

6 Выводы

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

Список литературы

1. Лабораторная работа №7
2. Таблица ASCII