

Лабораторная работа №6

Дисциплина: Архитектура компьютера

Первий Анастасия Андреевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	11
4.1	Символьные и численные данные в NASM	11
4.1.1	Программа вывода значения регистра eax(I)	11
4.1.2	Программа вывода значения регистра eax(II)	14
4.2	Выполнение арифметических операций в NASM	17
4.2.1	Программа вычисления выражения	17
4.2.2	Программа вычисления варианта задания по номеру студенческого билета	19
4.3	Задание для самостоятельной работы	20
5	Выводы	24
6	Листинги	25
6.1	Листинг 6.1. Программа вывода значения регистра eax	25
6.2	Листинг 6.2. Программа вывода значения регистра eax	25
6.3	Листинг 6.3. Программа вычисления выражения $x(x) = (5 \times 2 + 3)/3$	25
6.4	Листинг 6.4. Программа вычисления вычисления варианта задания по номеру студенческого билета	26
	Список литературы	27

Список иллюстраций

4.1	Создание каталога и файла	11
4.2	Текст программы вывода значения регистра eax(I)	12
4.3	Создание исполняемого файла и запуск программы	12
4.4	Текст программы вывода значения регистра eax(I). Измененный	13
4.5	Создание исполняемого файла и запуск программы	13
4.6	Создание файла	14
4.7	Текст программы вывода значения регистра eax(II)	14
4.8	Создание исполняемого файла и запуск программы	15
4.9	Текст программы вывода значения регистра eax(II). Измененный	15
4.10	Создание исполняемого файла и запуск программы	16
4.11	Текст программы вывода значения регистра eax(II). Измененный дважды	16
4.12	Создание исполняемого файла и запуск программы	17
4.13	Создание файла	17
4.14	Текст программы вычисления выражения	18
4.15	Создание исполняемого файла и запуск программы	18
4.16	Текст программы вычисления выражения. Измененный	19
4.17	Создание исполняемого файла и запуск программы	19
4.18	Создание файла	19
4.19	Текст программы вычисления варианта задания по номеру студен- ческого билета	20
4.20	Создание исполняемого файла и запуск программы	20
4.21	Код программы	22
4.22	Создание исполняемого файла и запуск программы	23

Список таблиц

1 Цель работы

Целью данной лабораторной работы является освоение арифметических инструкций языка ассемблера NASM.

2 Задание

0. Символьные и численные данные в NASM
1. Программа вывода значения регистра eax(I)
2. Программа вывода значения регистра eax(II)
3. Выполнение арифметических операций в NASM
4. Программа вычисления выражения
5. Программа вычисления вычисления варианта задания по номеру студенческого билета
6. Задание для самостоятельной работы

3 Теоретическое введение

Адресация в NASM

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.

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.

Арифметические операции в NASM. Целочисленное сложение **add**.

Схема команды целочисленного сложения **add** (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда **add** работает как с числами со знаком, так и без знака и выглядит следующим образом:

add,

Целочисленное вычитание **sub**. Команда целочисленного вычитания **sub** (от англ. subtraction – вычитание) работает аналогично команде **add** и выглядит следующим образом:

sub,

Так, например, команда `sub ebx,5` уменьшает значение регистра **ebx** на 5 и записывает результат в регистр **ebx**

Команды инкремента и декремента.

Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: **inc** (от англ. increment) и **dec** (от англ. decrement), которые увеличивают и уменьшают на 1 свой операнд. Эти команды содержат один операнд и имеет следующий вид:

inc dec

Операндом может быть регистр или ячейка памяти любого размера. Команды инкремента и декремента выгодны тем, что они занимают меньше места, чем соответствующие команды сложения и вычитания. Так, например, команда *inc ebx* увеличивает значение регистра **ebx** на 1, а команда *dec ax* уменьшает значение регистра **ax** на 1.

Команда изменения знака операнда **neg**.

Еще одна команда, которую можно отнести к арифметическим командам это команда изменения знака **neg**:

neg

Команда *neg* рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.

mov ax,1 ; AX = 1 neg ax ; AX = -1

Команды умножения **mul** и **imul**.

Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды. Для беззнакового умножения используется команда **mul** (от англ. multiply – умножение):

mul

Для знакового умножения используется команда **imul**:

imul

Для команд умножения один из сомножителей указывается в команде и должен находиться в регистре или в памяти, но не может быть непосредственным операндом. Второй сомножитель в команде явно не указывается и должен находиться в регистре **EAX,AX** или **AL**, а результат помещается в регистры **EDX:EAX, DX:AX** или **AX**, в зависимости от размера операнда.

Команды деления *div* и *idiv*.

Для деления, как и для умножения, существует 2 команды **div** (от англ. divide - деление) и **idiv**:

div ; Беззнаковое деление

idiv ; Знаковое деление

В командах указывается только один операнд – делитель, который может быть регистром или ячейкой памяти, но не может быть непосредственным операндом. Местоположение делимого и результата для команд деления зависит от размера делителя. Кроме того, так как в результате деления получается два числа – частное и остаток, то эти числа помещаются в определённые регистры.

Перевод символа числа в десятичную символьную запись

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 по-

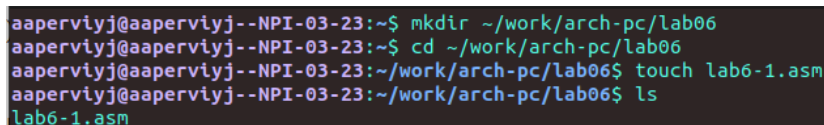
следовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно. Для выполнения лабораторных работ в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Это:

- `iprint` – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число (`mov eax, ...`).
- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки.
- `atoi` – функция преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число (`mov eax, ...`).

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

Перед тем, как начать выполнения лабораторной работы, необходимо создать каталог, в котором будут храниться необходимые файлы, а также сам файл **lab6-1.asm** (Рис. 4.1)



```
aaerviyj@aaerviyj--NPI-03-23:~$ mkdir ~/work/arch-pc/lab06
aaerviyj@aaerviyj--NPI-03-23:~$ cd ~/work/arch-pc/lab06
aaerviyj@aaerviyj--NPI-03-23:~/work/arch-pc/lab06$ touch lab6-1.asm
aaerviyj@aaerviyj--NPI-03-23:~/work/arch-pc/lab06$ ls
lab6-1.asm
```

Рис. 4.1: Создание каталога и файла

4.1.1 Программа вывода значения регистра **eax(I)**

Для выполнения данного пункта лабораторной работы требуется листинг 6.1. В нем находится текст программы вывода значения регистра **eax(I)**, который нужно ввести в файл **lab6-1.asm** (Рис. 2 4.2)



```
GNU nano 6.2 /home/aaperviyj/work/arch-pc/lab06/lab6-1.asm *
#include 'in_out.asm'

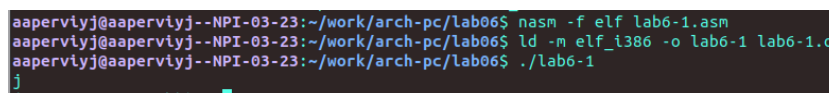
SECTION .bss
buf1: RESB 80

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. 4.2: Текст программы вывода значения регистра eax(I)

Создаю исполняемый файл и запускаю его (Рис.3 4.3)



```
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ./lab6-1
j
(reverse i search)~$
```

Рис. 4.3: Создание исполняемого файла и запуск программы

В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом будет символ j. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда add eax,ebx запишет в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j (см. таблицу ASCII в приложении).

Далее изменим текст программы и вместо символов, запишем в регистры числа. (Рис.4 4.4)

```
GLOBAL _start
_start:

mov  eax,6
mov  ebx,4
add  eax,ebx
mov  [buf1],eax
mov  eax,buf1
call sprintf
call quit
```

Рис. 4.4: Текст программы вывода значения регистра eax(I). Измененный

Создаю исполняемый файл и запускаю его (Рис.5 4.5)

```
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ./lab6-1

aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ █
```

Рис. 4.5: Создание исполняемого файла и запуск программы

Как и в предыдущем случае при исполнении программы я не получаю число 10.

В данном случае выводится символ с кодом 10. В соответствии с таблицей ASCII я определила, что у кода 10 нет символа. Поэтому программа вывела пустую строку.

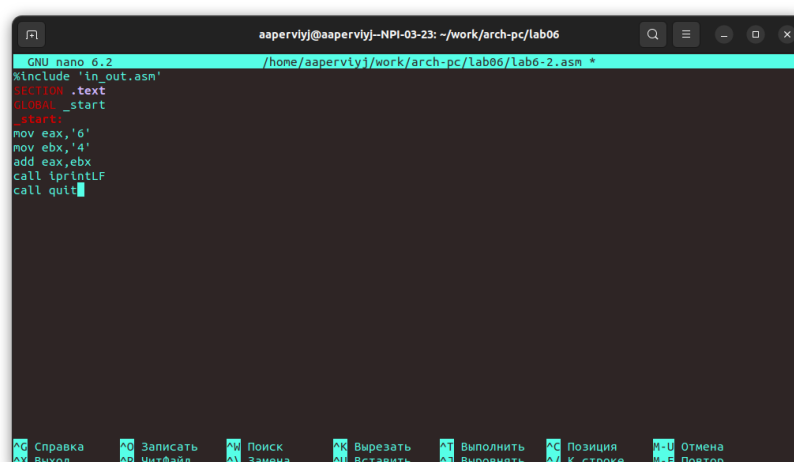
4.1.2 Программа вывода значения регистра `eax`(II)

Как отмечалось выше, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовываю текст программы из Листинга 6.1 с использованием этих функций.

Создаю файл **lab6-2.asm** в каталоге `~/work/arch-pc/lab06` (Рис.6 4.6) и ввожу в него текст программы из листинга 6.2 (Рис.7 4.7)

```
aaperviyj@aaperviyj--NPI-03-23: ~/work/arch-pc/lab06$ touch lab6-2.asm
aaperviyj@aaperviyj--NPI-03-23: ~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2.asm
```

Рис. 4.6: Создание файла



```
GNU nano 6.2 /home/aaperviyj/work/arch-pc/lab06/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 вывода значения регистра `eax`(II)

Ввела текст программы, теперь нужно создать исполняемый файл и запустить его (Рис.8 4.8)

```

aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ./lab6-2
106
(reverse + search)`'

```

Рис. 4.8: Создание исполняемого файла и запуск программы

В результате работы программы я получила число 106. В данном случае, как и в первом, команда `add` складывает коды символов '6' и '4' ($54+52=106$). Однако, в отличие от программы из листинга 6.1, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменю символы на числа. (Рис. 9 4.9)

```

#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: Текст программы вывода значения регистра `eax`(II). Измененный

Снова создаю исполняемый файл и запускаю его. (Рис.10 4.10)

```

aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ./lab6-2
10
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$

```

Рис. 4.10: Создание исполняемого файла и запуск программы

В результате работы программы я получила число 10. В данном случае, команда add складывает числа 6 и 4 ($6+4=10$). Теперь заменяю функцию `iprintLF` на `iprint`. (Рис.11 4.11) Создаю исполняемый файл и запускаю его. (Рис.12 4.12)

```

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

```

Рис. 4.11: Текст программы вывода значения регистра `eax`(II). Измененный дважды


```

aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ./lab6-2
10aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$

```

Рис. 4.12: Создание исполняемого файла и запуск программы

Вывод функций `iprintLF` и `iprint` отличается тем, что результат в первом случае выводится на отдельной строке. Во втором же случае число выводится на одной строке со строкой ввода.

4.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приводится программа вычисления арифметического выражения $\boxtimes(\boxtimes) = (5 \boxtimes 2 + 3)/3$.

Для выполнения этого пункта необходимо создать файл **lab6-3.asm** в каталоге `~/work/arch-pc/lab06` (Рис.13 4.13)

```

10aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ touch lab6-3.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6-2.o lab6-3.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$

```

Рис. 4.13: Создание файла

4.2.1 Программа вычисления выражения

Ввожу текст программы из листинга 6.3 в файл **lab6-3.asm** (Рис.14 4.14)

```

GNU nano 6.2 /home/aaperviyj/work/arch-pc/lab06/lab6-3.asm
; 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 tprintf ; 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 tprintf ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 4.14: Текст программы вычисления выражения

Создаю исполняемый файл и запускаю его (Рис.15 4.15)

```

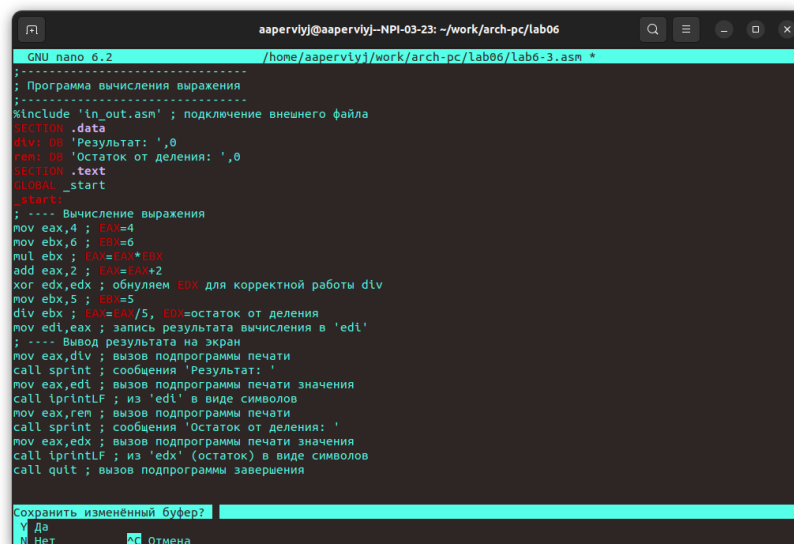
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$

```

Рис. 4.15: Создание исполняемого файла и запуск программы

Результат работы программы должен был быть следующим: user@dk4n31:~\$./lab6-3 Результат: 4 Остаток от деления: 1

Такой результат я и получила. Теперь я могу изменить текст программы для вычисления выражения $\text{X}(\text{X}) = (4 \times 6 + 2)/5$ (Рис.16 4.16)

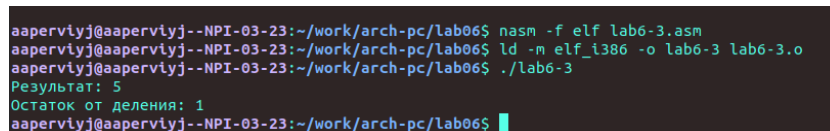


```
GNU nano 6.2 /home/aaperviyj/work/arch-pc/lab06/lab6-3.asm *
;-----
; Программа вычисления выражения
;-----
%include 'in_out.asm' ; подключение внешнего файла
section .data
div: db 'Результат: ',0
rem: db 'Остаток от деления: ',0
section .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4 ; EDI=4
mov ebx,0 ; EDI=6
mul ebx ; EAX=EAX*EDI
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDI для корректной работы div
mov ebx,5 ; EDI=5
div ebx ; EAX=EAX/5, EDI=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 tprintf ; 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 tprintf ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения

Сохранить изменённый буфер?
Y Да
N Нет  Отмена
```

Рис. 4.16: Текст программы вычисления выражения. Измененный

Снова создаю исполняемый файл и запускаю его (Рис.17 4.17)

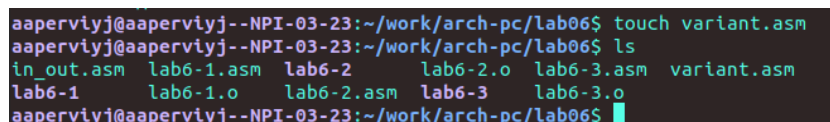


```
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$
```

Рис. 4.17: Создание исполняемого файла и запуск программы

4.2.2 Программа вычисления варианта задания по номеру студенческого билета

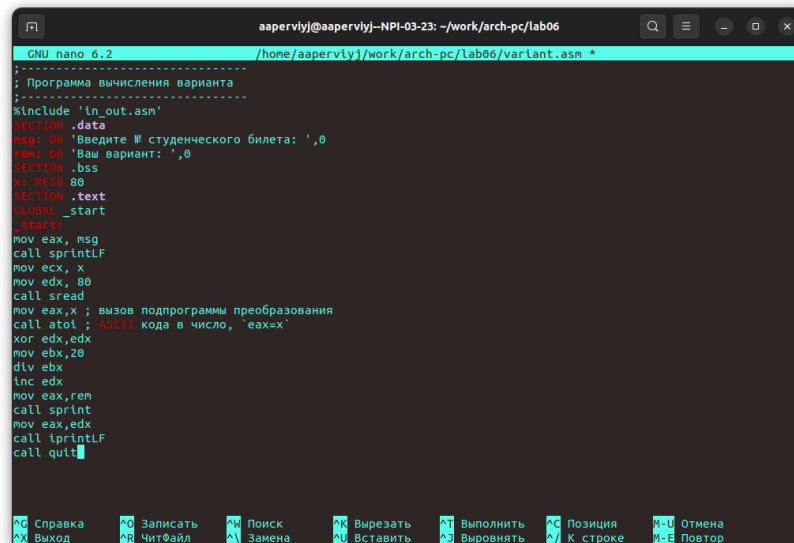
В этой программе входные данные будут вводиться с клавиатуры. Создаю файл **variant.asm** в каталоге **~/work/arch-pc/lab06** (Рис.18 4.18)



```
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ touch variant.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1.asm lab6-2 lab6-2.o lab6-3.asm variant.asm
lab6-1 lab6-1.o lab6-2.asm lab6-3 lab6-3.o
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$
```

Рис. 4.18: Создание файла

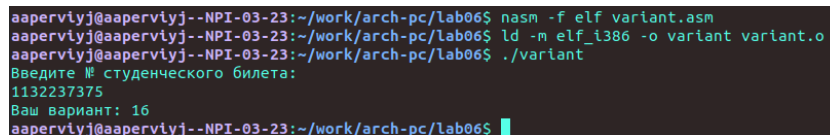
Теперь нужно ввести код программы из листинга 6.4 (Рис.19 4.19)



```
GNU nano 6.2 /home/aaperviyj/work/arch-pc/lab06/variant.asm *
;-----
; 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 sprintf
mov eax, edx
call iprintf
call quit
```

Рис. 4.19: Текст программы вычисления варианта задания по номеру студенческого билета

Создаю исполняемый файл и запускаю программу (Рис.20 4.20)



```
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ nasm -f elf variant.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132237375
Ваш вариант: 16
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$
```

Рис. 4.20: Создание исполняемого файла и запуск программы

Когда программа запросила № студенческого билета, я ввела свой. Результатом был 16 вариант

4.3 Задание для самостоятельной работы

Попробую написать программу, которая вычислит выражение. Мне выпал 16 вариант, поэтому мне необходимо написать программу, которая вычислит

следующее выражение $(10x - 5)^2$, число x должно вводиться пользователем. Вот программа, которая яу меня получилась(Рис.21 4.22):

```
%include 'in_out.asm'
SECTION .data msg: DB 'Введите X:',0 rem: DB 'Результат:',0
SECTION .bss x: RESB 80
SECTION .text GLOBAL _start _start:
; -- Вычисление выражения mov eax, msg call sprint mov ecx, x mov edx, 80 call
sread mov eax,x call atoi; ASCII кода в число, 'eax=x' mov ebx, 10 mul ebx add eax, -5
mov ebx, eax mul ebx mov edi, eax
mov eax, rem call sprint mov eax,edi call iprintLF
call quit
```

```

%include      'in_out.asm'

SECTION .data
msg: DB 'Введите X: ',0
rem: DB 'Результат: ',0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

; ---- Вычисление выражения
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x
call atoi; ASCII кода в число, 'eax=x'
mov ebx, 10
mul ebx
add  eax, -5
mov ebx, eax
mul ebx
mov edi, eax

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

call quit

```

Рис. 4.21: Код программы

Теперь нужно проверить корректность этой программы (Рис.22 ??)

```
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ nasm -f elf zadavie.asm
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ld -m elf_i386 -o zadavie.o zadavie.o
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$ ./zadavie
Введите X: 3
Результат: 625
aaperviyj@aaperviyj--NPI-03-23:~/work/arch-pc/lab06$
```

Рис. 4.22: Создание исполняемого файла и запуск программы

Проверяем программу арифметически: $(10*3-5)^{2=25} 2=625$

Убедились, что программа работает кореектно, значит задание выполнено.

5 Выводы

Во время выполнения лабораторной работы я научилась использовать арифметические инструкции языка ассемблера NASM.

6 Листинги

6.1 Листинг 6.1. Программа вывода значения регистра еах

```
%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
```

6.2 Листинг 6.2. Программа вывода значения регистра еах

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

6.3 Листинг 6.3. Программа вычисления выражения $(5 \times 2 + 3)/3$

```
;----- ; 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`

6.4 Листинг 6.4. Программа вычисления вычисления варианта задания по номеру студенческого билета

```
;----- ; Программа вычисления варианта ;-----  
%include 'in_out.asm'  
SECTION .data msg: DB 'Введите № студенческого билета:',0 rem: DB 'Ваш вари-  
ант:',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  
mov eax,rem call sprint mov eax,edx call iprintLF  
call quit
```

Список литературы

Архитектура ЭВМ