

Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Буриева Шахзода Акмаловна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	8
4	Выполнение заданий для самостоятельной работы	16
5	Ответы на вопросы.	19
6	Выводы	21
	Список литературы	22

Список иллюстраций

3.1	Создание каталога и файла в нём	8
3.2	Копирование файла in_out.asm в каталог	9
3.3	Ввод текста программы	9
3.4	Создание исполняемого файла и его запуск	10
3.5	Изменение программы	10
3.6	Создание исполняемого файла и его запуск	10
3.7	Создание файла lab6-2.asm	11
3.8	Ввод текста программы	11
3.9	Создание исполняемого файла и его запуск	11
3.10	Изменение текста программы	12
3.11	Создание исполняемого файла lab6-2.asm и его запуск	12
3.12	Замена на iprint	12
3.13	Создание исполняемого файла lab6-2.asm и его запуск	13
3.14	Создание файла lab6-3.asm	13
3.15	Ввод текста программы в файл lab6-3.asm	13
3.16	Создание исполняемого файла lab6-3.asm и его запуск	14
3.17	Изменение текста программы для алгебраического вычисления	14
3.18	Создание изменённого исполняемого файла для алгебраического вычисления и его запуск	14
3.19	Создание файла variant.asm и его запуск	15
3.20	Ввод текста программы в файл variant.asm	15
3.21	Создание исполняемого файла variant.asm и его запуск	15
4.1	Создание файла lab6-4.asm	16
4.2	Ввод текста программы в файл lab6-4.asm	17
4.3	Текст моей программы в редакторе	17
4.4	Создание исполняемого файла lab6-3.asm и его запуск	18
5.1	Код для вывода сообщения “Ваш вариант”	19
5.2	Код для вычисления варианта	20
5.3	Код для вывода результатов	20

Список таблиц

1 Цель работы

Освоить арифметических инструкций языка ассемблера NASM.

2 Теоретическое введение

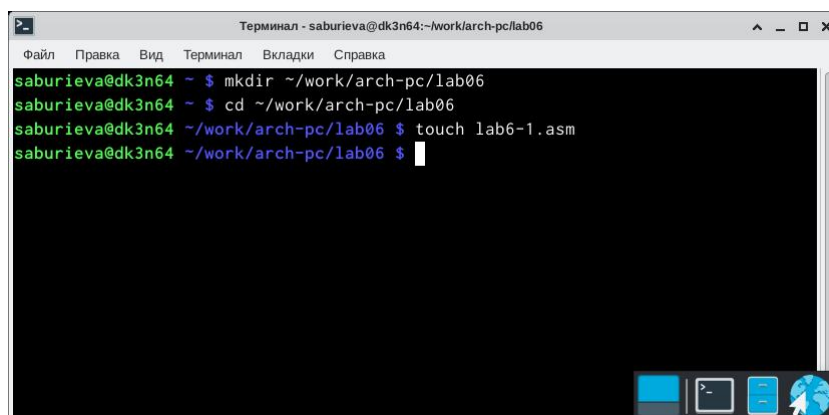
Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Схема команды целочисленного сложения `add` (от англ. `addition` - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. 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.

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 вы-

водить 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.

3 Выполнение лабораторной работы

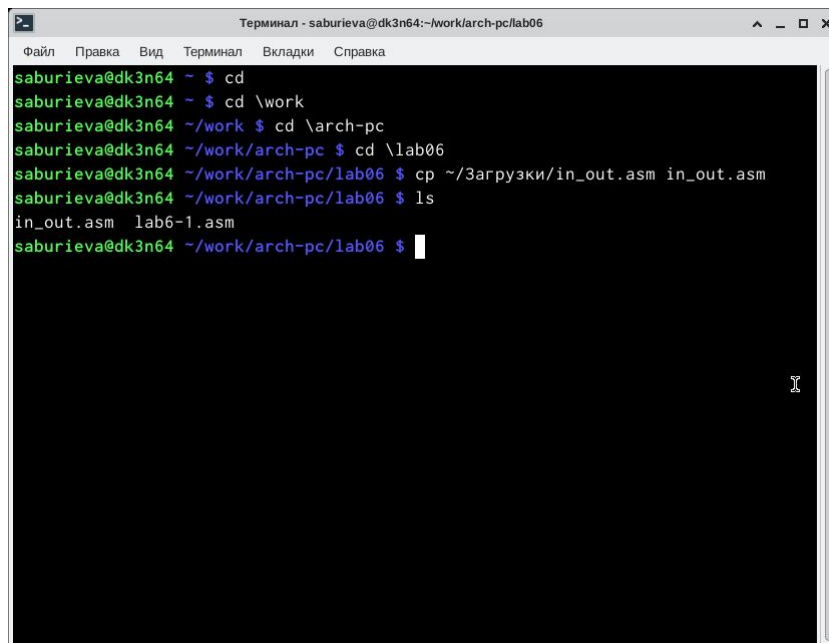
Создала каталог для программ лабораторной работы № 6, перешла в него и создала файл lab6-1.asm



```
Терминал - saburieva@dk3n64:~/work/arch-pc/lab06
Файл  Правка  Вид  Терминал  Вкладки  Справка
saburieva@dk3n64 ~ $ mkdir ~/work/arch-pc/lab06
saburieva@dk3n64 ~ $ cd ~/work/arch-pc/lab06
saburieva@dk3n64 ~/work/arch-pc/lab06 $ touch lab6-1.asm
saburieva@dk3n64 ~/work/arch-pc/lab06 $
```

Рис. 3.1: Создание каталога и файла в нём

Перед созданием исполняемого файла создала копию файла in_out.asm в каталоге ~/work/arch-pc/lab06.

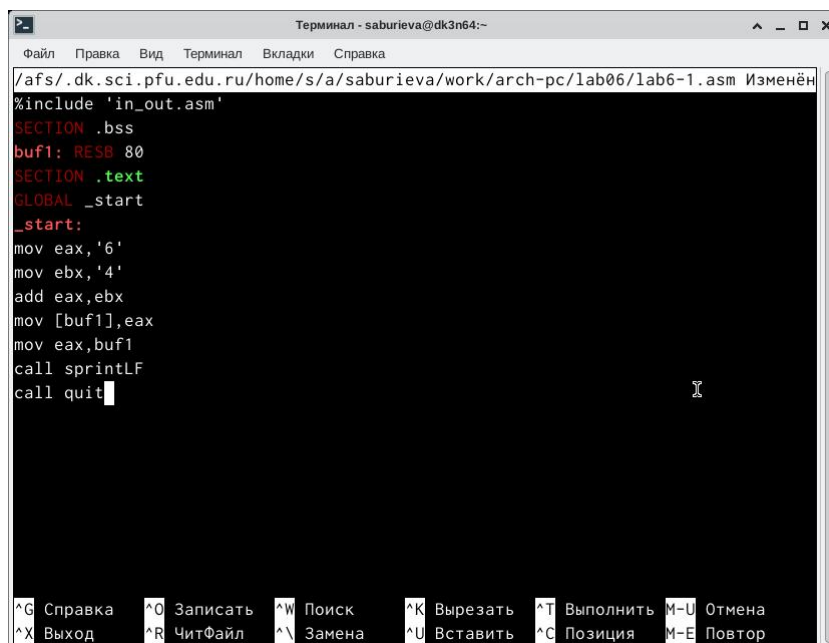


A terminal window titled "Терминал - saburieva@dk3n64:~/work/arch-pc/lab06". The window contains the following commands and output:

```
saburieva@dk3n64 ~ $ cd
saburieva@dk3n64 ~ $ cd \work
saburieva@dk3n64 ~/work $ cd \arch-pc
saburieva@dk3n64 ~/work/arch-pc $ cd \lab06
saburieva@dk3n64 ~/work/arch-pc/lab06 $ cp ~/Загрузки/in_out.asm in_out.asm
saburieva@dk3n64 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1.asm
saburieva@dk3n64 ~/work/arch-pc/lab06 $
```

Рис. 3.2: Копирование файла in_out.asm в каталог

Ввела в файл lab6-1.asm текст программы из данного листинга и сохранила его.



A terminal window titled "Терминал - saburieva@dk3n64:~". The window shows the path to a file and its content:

```
/afs/.dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab06/lab6-1.asm Изменён
#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
```

At the bottom of the window, there is a menu bar with the following options:

^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить	M-U Отмена
^X Выход	^R ЧитФайл	^N Замена	^U Вставить	^C Позиция	M-E Повтор

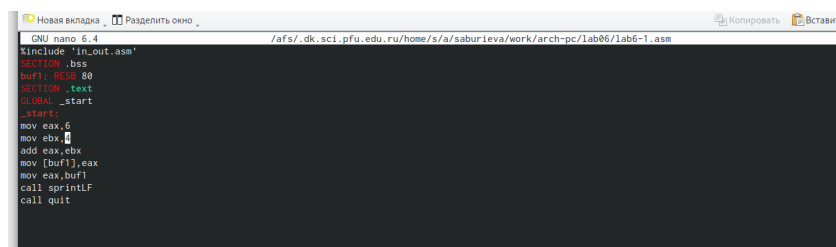
Рис. 3.3: Ввод текста программы

Создала исполняемый файл и запустила его. Программа вывела символ j, потому что он соответствует сумме двоичных кодов символов 4 и 6 по системе ASCII.

```
saburieva@dk5n52 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
saburieva@dk5n52 ~/work/arch-pc/lab06 $ d -m elf_i386 -o lab6-1 lab6-1.o
bash: d: команда не найдена
saburieva@dk5n52 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
saburieva@dk5n52 ~/work/arch-pc/lab06 $ ./lab6-1
j
saburieva@dk5n52 ~/work/arch-pc/lab06 $
```

Рис. 3.4: Создание исполняемого файла и его запуск

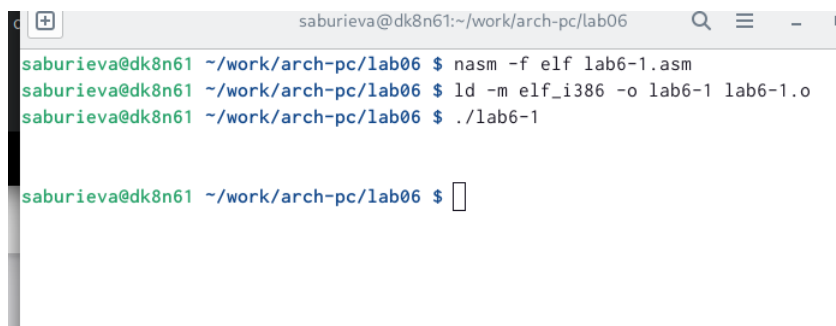
Изменила текст программы.



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/s/a/saburieva/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, 1
add eax, ebx
mov [buf1], eax
mov eax, buf1
call sprintf
call printf
call quit
```

Рис. 3.5: Изменение программы

Создала изменённый исполняемый файл и запустила его. Программа вывела символ перевода строки, потому что он соответствует коду 10=6+4 по системе ASCII.

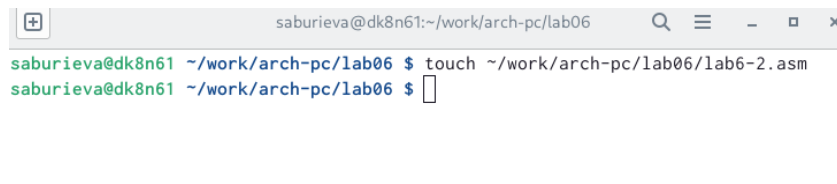


```
saburieva@dk8n61 ~/work/arch-pc/lab06
saburieva@dk8n61 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
saburieva@dk8n61 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
saburieva@dk8n61 ~/work/arch-pc/lab06 $ ./lab6-1

saburieva@dk8n61 ~/work/arch-pc/lab06 $
```

Рис. 3.6: Создание исполняемого файла и его запуск

Создала файл lab6-2.asm в каталоге ~/work/arch-pc/lab06.

A terminal window with the title bar 'saburieva@dk8n61:~/work/arch-pc/lab06'. The prompt is 'saburieva@dk8n61 ~/work/arch-pc/lab06 \$'. The command 'touch ~/work/arch-pc/lab06/lab6-2.asm' has been entered and executed. The prompt is now 'saburieva@dk8n61 ~/work/arch-pc/lab06 \$' with a cursor.

```
saburieva@dk8n61:~/work/arch-pc/lab06
saburieva@dk8n61 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
saburieva@dk8n61 ~/work/arch-pc/lab06 $
```

Рис. 3.7: Создание файла lab6-2.asm

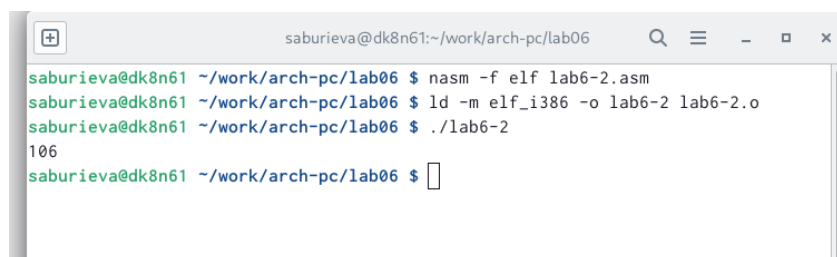
Ввела в созданный файл lab6-2.asm текст программы из листинга.

A terminal window with the title bar 'mc [saburieva@dk8n61]:~/work/arch-pc/lab06'. The prompt is 'mc [saburieva@dk8n61]:~/work/arch-pc/lab06'. The file path '/afs/.dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab06/lab6-2.asm' is shown. The content of the file is displayed:

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

Рис. 3.8: Ввод текста программы

Создала исполняемый файл и запустила его. Программа вывела, а именно сложение кодов символов “6” и “4”. В итоге получилось 106.

A terminal window with the title bar 'saburieva@dk8n61:~/work/arch-pc/lab06'. The prompt is 'saburieva@dk8n61 ~/work/arch-pc/lab06 \$'. The commands and their outputs are:

```
saburieva@dk8n61 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
saburieva@dk8n61 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
saburieva@dk8n61 ~/work/arch-pc/lab06 $ ./lab6-2
106
saburieva@dk8n61 ~/work/arch-pc/lab06 $
```

Рис. 3.9: Создание исполняемого файла и его запуск

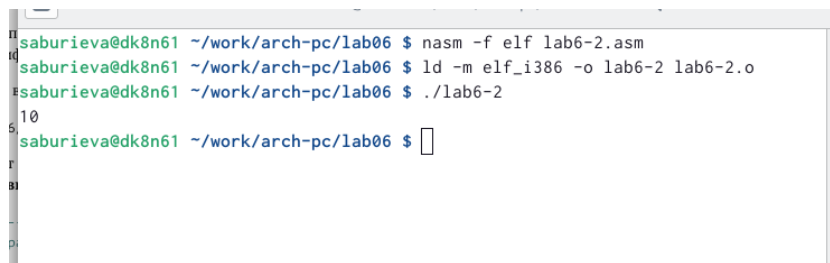
Далее опять изменила текст программы lab6-2.asm.



```
1 /afs/.dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab06/lab6-2.asm Изменён
2 %include 'in_out.asm'
3 SECTION .text
4 GLOBAL _start
5 _start:
6 mov eax,6
7 mov ebx,4
8 add eax,ebx
9 call iprintLF
10 call quit
```

Рис. 3.10: Изменение текста программы

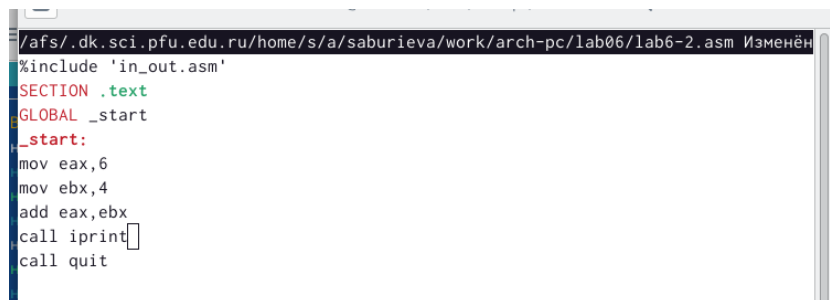
Создала изменённый исполняемый файл lab6-2.asm и запустила его. Программа вывела сложение именно цифр.



```
13 saburieva@dk8n61 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
14 saburieva@dk8n61 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
15 saburieva@dk8n61 ~/work/arch-pc/lab06 $ ./lab6-2
16 10
17 saburieva@dk8n61 ~/work/arch-pc/lab06 $
```

Рис. 3.11: Создание исполняемого файла lab6-2.asm и его запуск

Заменяла функцию iprintLF на iprint в данном файле.



```
1 /afs/.dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab06/lab6-2.asm Изменён
2 %include 'in_out.asm'
3 SECTION .text
4 GLOBAL _start
5 _start:
6 mov eax,6
7 mov ebx,4
8 add eax,ebx
9 call iprint
10 call quit
```

Рис. 3.12: Замена на iprint

Создала изменённый исполняемый файл lab6-2.asm и запустила его. Вывод остался таким же, потому что символьный перенос строки не отображался, когда

программа выполнялась с ограничением `iprintLF`, а `iprint` не добавлял к завершению символный перенос строк, в отличие от `iprintLF`.

```
saburieva@dk8n61 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
saburieva@dk8n61 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
saburieva@dk8n61 ~/work/arch-pc/lab06 $ ./lab6-2
10saburieva@dk8n61 ~/work/arch-pc/lab06 $
```

Рис. 3.13: Создание исполняемого файла `lab6-2.asm` и его запуск

Создала файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06`.

```
saburieva@dk8n61 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm
saburieva@dk8n61 ~/work/arch-pc/lab06 $
```

Рис. 3.14: Создание файла `lab6-3.asm`

Внимательно изучила текст программы из листинга и ввела в `lab6-3.asm`.

```
GNU nano 6.4 ~/work/arch-pc/lab06/lab6-3.asm
#include "in_out.asm" ; подключение внешнего файла
SECTION .data
    edi: DB 'Результат: ',0
    edx: 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
    dd eax,3 ; EAX=EAX*3
    or edx,edx ; обнулен EDX для корректной работы div
    mov ebx,3 ; EBX=3
    div ebx ; EAX=EAX/3, EDX=остаток от деления
    mov edi,edx ; запись результата вычисления в 'edi'
    ---- Вывод результата на экран
    mov eax,edi ; вызов подпрограммы печати
    all: print ; сообщения 'Результат: '
    mov eax,edi ; вызов подпрограммы печати значения
    all: iprintLF ; из 'edi' в виде символов
    mov eax,edx ; вызов подпрограммы печати
    all: print ; сообщения 'Остаток от деления: '
    mov eax,edx ; вызов подпрограммы печати значения
    all: iprintLF ; из 'edx' (остаток) в виде символов
    all: quit ; вызов подпрограммы завершения
```

Рис. 3.15: Ввод текста программы в файл `lab6-3.asm`

Создала исполняемый файл и запустила его. На экран вывелся ответ.

```
saburueva@dk8n61 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
saburueva@dk8n61 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
saburueva@dk8n61 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
saburueva@dk8n61 ~/work/arch-pc/lab06 $
```

Рис. 3.16: Создание исполняемого файла lab6-3.asm и его запуск

Изменила текст программы для вычисления выражения $4 \times (6 + 2) / 5$.

```
GNU nano 2.4.0 /afs/dk.sci.pfu.edu.ru/home/s/a/saburueva/work/arch-pc/lab06/lab6-3.asm
; ---- Вычисление выражения
mov eax,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EBX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDI для корректной работы div
mov ebx,5 ; EBX=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 sprintf ; сообщения "Результат: "
mov eax,edi ; вызов подпрограммы печати значения
call iprintf ; из "edi" в виде символов
mov eax,edx ; вызов подпрограммы печати
call sprintf ; сообщения "Остаток от деления: "
mov eax,edx ; вызов подпрограммы печати значения
call iprintf ; из "edx" (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 3.17: Изменение текста программы для алгебраического вычисления

Создала изменённый исполняемый файл для алгебраического вычисления и запустила его.

```
saburueva@dk8n61 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
saburueva@dk8n61 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
saburueva@dk8n61 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
saburueva@dk8n61 ~/work/arch-pc/lab06 $
```

Рис. 3.18: Создание изменённого исполняемого файла для алгебраического вычисления и его запуск

Создала файл variant.asm в каталоге ~/work/arch-pc/lab06.

```
saburieva@dk8n61 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
saburieva@dk8n61 ~/work/arch-pc/lab06 $
```

Рис. 3.19: Создание файла variant.asm и его запуск

Внимательно изучила текст программы из листинга и ввела в файл variant.asm.

```
GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab06/variant.asm
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 00
call read
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, ren
call sprintf
mov eax, edx
call iprintf
call quit
```

Рис. 3.20: Ввод текста программы в файл variant.asm

Создала файл variant.asm и запустила его. На запрос ввести студенческий билет, ввела номер своего студенческого билета. И мне выпало число 4.

```
saburieva@dk2n26 ~/work/arch-pc/lab06 $ touch lab6-3.asm
saburieva@dk2n26 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132236103
Ваш вариант: 4
saburieva@dk2n26 ~/work/arch-pc/lab06 $
```

Рис. 3.21: Создание исполняемого файла variant.asm и его запуск

4 Выполнение заданий для самостоятельной работы

Создала файл lab6-4.asm в каталоге ~/work/arch-pc/lab06.



```
saburieva@dk8n61 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-4.asm
saburieva@dk8n61 ~/work/arch-pc/lab06 $
```

Рис. 4.1: Создание файла lab6-4.asm

Ввела текст программы для алгебраического расчёта.


```

/afs/.dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab06/lab6-4.asm
#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 неинициализированных данных
x: RESB 80 ; Переменная, значение которой будем вводить с клавиатуры, выделенный р>
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'
sub eax, 1;
mov ebx, 4
mul ebx;

```

Рис. 4.2: Ввод текста программы в файл lab6-4.asm

На рисунке показан сам текст моей программы, который я вводила.

```

#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 неинициализированных данных
x: RESB 80 ; Переменная, значение которой будем вводить с клавиатуры, выделенный размер - 80 байтов
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'
sub eax, 1;
mov ebx, 4
mul ebx;
mov ebx, 3
div ebx;
add eax, 5;
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 iprintf ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 4.3: Текст моей программы в редакторе

Создала исполняемый файл и запустила его. На запрос “Введите значение переменной x” ввела число 4. А затем ввела число 10. На экран вывелись оба значения X.

```
saburieva@dk2n22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
saburieva@dk2n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
saburieva@dk2n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
saburieva@dk2n22 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 4
Результат: 9
saburieva@dk2n22 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 10
Результат: 17
saburieva@dk2n22 ~/work/arch-pc/lab06 $
```

Рис. 4.4: Создание исполняемого файла lab6-3.asm и его запуск

5 Ответы на вопросы.

- 1) Для вывода сообщения “Ваш вариант” предоставляются следующие строки кода.

```
mov  eax, msg  
call sprintf
```

Рис. 5.1: Код для вывода сообщения “Ваш вариант”

2) Инструкция `mov ecx, x` используется для того чтобы поставить адрес вводимой строки `x` в регистр, `ecx mov edx, 80` - это запись в регистр `edx` длины вводимой строки, `call sread` - это вызов подпрограммы из внешнего файла, вводы сообщений с клавиатуры.

3) Инструкция “`call atoi`” используется для вызова подпрограмм из внешнего файла, преобразующий символы ASCII-кода в имена чисел и записывающая результат в регистр `eax`.

4) Эти строки листинга 6.4 отвечают за вычисления варианта.

```
xor edx,edx
mov ebx,20
div ebx
inc edx
```

Рис. 5.2: Код для вычисления варианта

5) При выполнении инструкции остаток деления `div ebx` записывается в регистр `edx`.

б) Инструкция “inc edx” используется для увеличения значения регистра edx на 1.

7) Для вывода результатов на экране используются эти коды.

```
mov     eax,edx
call    iprintLF
```

Рис. 5.3: Код для вывода результатов

Отправила все файлы на github.

6 Выводы

Я освоила арифметических инструкций языка ассемблера NASM.

Список литературы