

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Киселева Елизавета Александровна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Символьные и численные данные в NASM	8
4.2	Выполнение арифметических операций в NASM	14
4.2.1	Ответы на вопросы по программе	17
4.3	Выполнение заданий для самостоятельной работы	18
5	Выводы	21
6	Список литературы	22

Список иллюстраций

4.1	Создание директории и файла	8
4.2	Редактирование файла	9
4.3	Запуск исполняемого файла	9
4.4	Редактирование файла	10
4.5	Запуск исполняемого файла	11
4.6	Создание файла	11
4.7	Редактирование файла	11
4.8	Запуск исполняемого файла	12
4.9	Редактирование файла	12
4.10	Запуск исполняемого файла	13
4.11	Редактирование файла	13
4.12	Запуск исполняемого файла	13
4.13	Создание файла	14
4.14	Редактирование файла	14
4.15	Запуск исполняемого файла	15
4.16	Изменение программы	15
4.17	Запуск исполняемого файла	16
4.18	Создание файла	16
4.19	Редактирование файла	17
4.20	Запуск исполняемого файла	17
4.21	Написание программы	19
4.22	Запуск исполняемого файла	19

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. - 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.

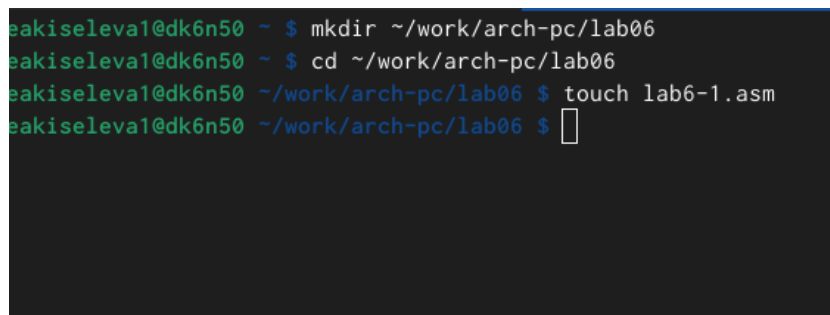
Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. 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

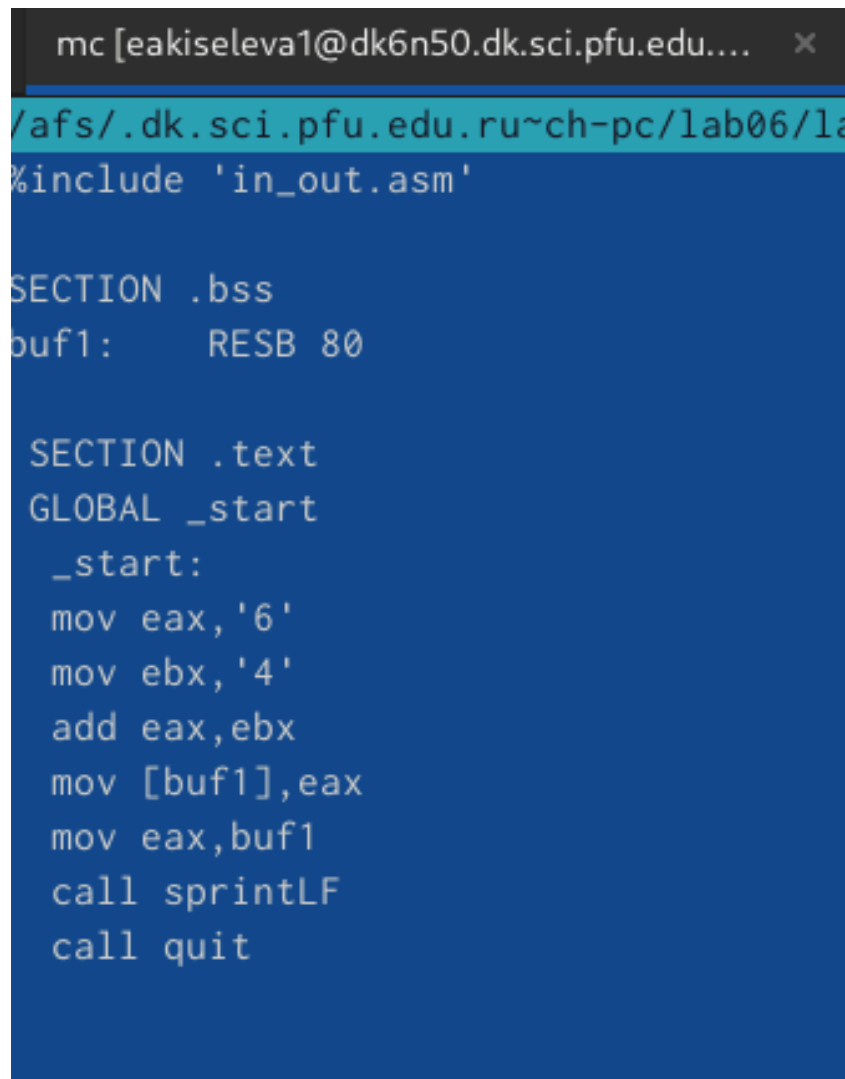
С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6. Перехожу в созданный каталог с помощью утилиты `cd`. С помощью утилиты `touch` создаю файл `lab6-1.asm` (рис. 4.1).



```
eakiseleva1@dk6n50 ~ $ mkdir ~/work/arch-pc/lab06
eakiseleva1@dk6n50 ~ $ cd ~/work/arch-pc/lab06
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ touch lab6-1.asm
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $
```

Рис. 4.1: Создание директории и файла

Скачиваю в текущий каталог файл `in_out.asm`, т.к. он будет использоваться в других программах. Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 4.2).



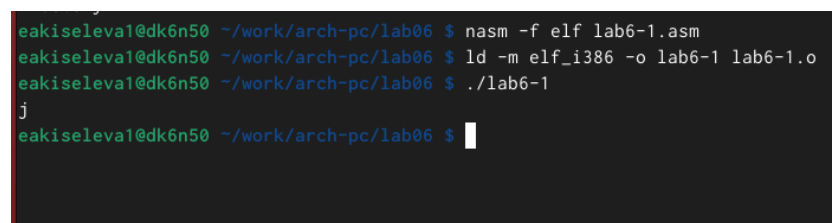
```
mc [eakiseleva1@dk6n50.dk.sci.pfu.edu.... x
/afs/.dk.sci.pfu.edu.ru~ch-pc/lab06/lab6-1.asm
#include 'in_out.asm'

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: Редактирование файла

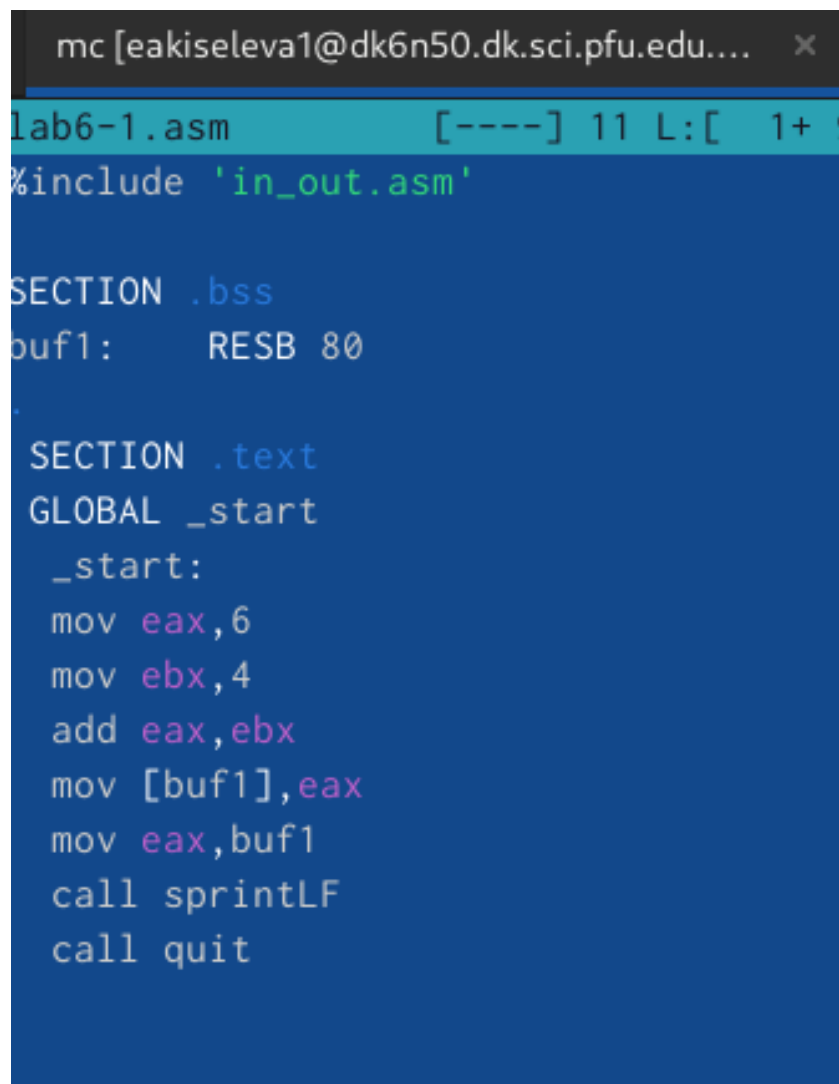
Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6 (рис. 4.3).



```
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ ./lab6-1
j
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $
```

Рис. 4.3: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.4).



```
mc [eakiseleva1@dk6n50.dk.sci.pfu.edu... x
lab6-1.asm [----] 11 L:[ 1+ 9
%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: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его. Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран (рис. 4.5).

```
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ ./lab6-1

eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $
```

Рис. 4.5: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 4.6).

```
eakiseleva1@dk6n50 ~ $ touch ~/work/arch-pc/lab06/lab6-2.asm
eakiseleva1@dk6n50 ~ $ nasm -f elf lab6-2.asm
```

Рис. 4.6: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 4.7).

```
~/work/arch-
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. 4.7: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2. Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4” (рис. 4.8).

```

eakiseleva1@dk6n50 ~ $ cd ~/work/arch-pc/lab06
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ ./lab6-2
106
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $

```

Рис. 4.8: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 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: Редактирование файла

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод

10 (рис. 4.10).

```
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ ./lab6-2
10
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $
```

Рис. 4.10: Запуск исполняемого файла

Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. 4.11).

```
~/work/arch-pc/lab06
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. 4.11: Редактирование файла

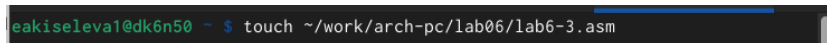
Создаю и запускаю новый исполняемый файл. Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией `iprintLF`, а `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF` (рис. 4.12).

```
10
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $ ./lab6-2
10eakiseleva1@dk6n50 ~/work/arch-pc/lab06 $
```

Рис. 4.12: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

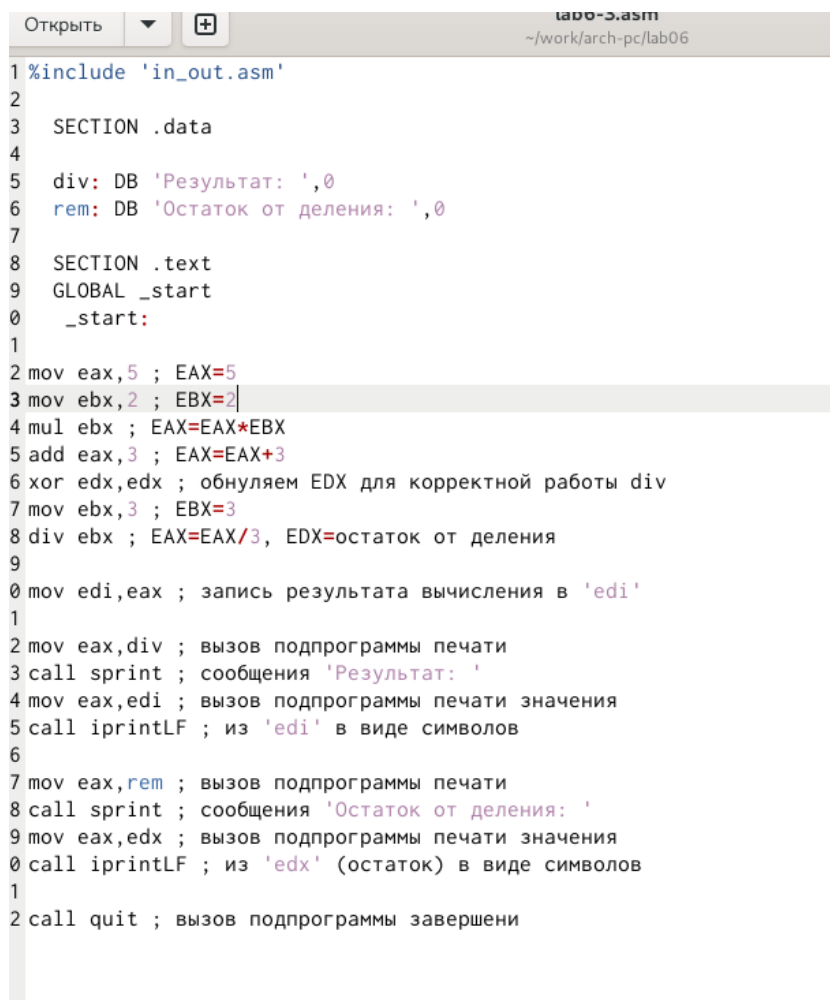
Создаю файл lab6-3.asm с помощью утилиты touch (рис. 4.13).



```
eakiseleva1@dk6n50 ~ $ touch ~/work/arch-pc/lab06/lab6-3.asm
```

Рис. 4.13: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 4.14).



```
Открыть  lab6-3.asm
~/work/arch-pc/lab06
1 %include 'in_out.asm'
2
3 SECTION .data
4
5 div: DB 'Результат: ',0
6 rem: DB 'Остаток от деления: ',0
7
8 SECTION .text
9 GLOBAL _start
0 _start:
1
2 mov eax,5 ; EAX=5
3 mov ebx,2 ; EBX=2
4 mul ebx ; EAX=EAX*EBX
5 add eax,3 ; EAX=EAX+3
6 xor edx,edx ; обнуляем EDX для корректной работы div
7 mov ebx,3 ; EBX=3
8 div ebx ; EAX=EAX/3, EDX=остаток от деления
9
0 mov edi,eax ; запись результата вычисления в 'edi'
1
2 mov eax,div ; вызов подпрограммы печати
3 call sprint ; сообщения 'Результат: '
4 mov eax,edi ; вызов подпрограммы печати значения
5 call iprintLF ; из 'edi' в виде символов
6
7 mov eax,rem ; вызов подпрограммы печати
8 call sprint ; сообщения 'Остаток от деления: '
9 mov eax,edx ; вызов подпрограммы печати значения
0 call iprintLF ; из 'edx' (остаток) в виде символов
1
2 call quit ; вызов подпрограммы завершения
```

Рис. 4.14: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.15).

```

eakiseval@dk6n50 ~ $ cd ~/work/arch-pc/lab06
eakiseval@dk6n50 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
eakiseval@dk6n50 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
eakiseval@dk6n50 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
eakiseval@dk6n50 ~/work/arch-pc/lab06 $

```

Рис. 4.15: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.16).

```

/afs/.dk.sci.pfu.edu.ru~ch-pc/lab06/lab6-3.asm 1105/
#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0

SECTION .text
GLOBAL _start
_start:

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'

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 завершени

```

Рис. 4.16: Изменение программы

Создаю и запускаю новый исполняемый файл. Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно (рис. 4.17).

```
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 4.17: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 4.18).

```
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
```

Рис. 4.18: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.19).


```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg: DB 'Введите № студенческого билета: ',0
5 rem: DB 'Ваш вариант: ',0
6
7 SECTION .bss
8 x:   RESB 80
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13
14 mov eax, msg
15 call sprintLF
16
17 mov ecx, x
18 mov edx, 80
19 call sread
20
21 mov eax, x ; вызов подпрограммы преобразования
22 call atoi ; ASCII кода в число, 'eax=x' xor edx,edx
23
24 xor edx,edx
25 mov ebx,20
26 div ebx
27 inc edx
28
29 mov eax,rem
30 call sprint
31 mov eax,edx
32 call iprintLF
33
34 call quit

```

Рис. 4.19: Редактирование файла

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 16 (рис. 4.20).

```

mc [eakiseleva1@dk3n... x eakiseleva1@dk3n55 - L... x eakiseleva1@dk3n55 - L... x
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132246735
Ваш вариант: 16
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $

```

Рис. 4.20: Запуск исполняемого файла

4.2.1 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
```

4.3 Выполнение заданий для самостоятельной работы

Создаю файл `lab7-4.asm` с помощью утилиты `touch`. Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(10 \cdot x - 5)^2$. Это выражение было под вариантом 16 (рис. 4.21).

```

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной 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
mov ebx, 10 ; EBX=10
mul ebx ; EAX=EAX*10
add eax, -5 ; EAX=EAX*10-5
mul eax ; EAX=EAX*EAX
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 iprintLF ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 4.21: Написание программы

Создаю и запускаю исполняемый файл. При вводе значения 1 , вывод - 25, при вводе значения 3 , вывод - 625. Вычисления выполнены верно (рис. 4.22).

```

25eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 1
Результат: 25
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 3
Результат: 625
eakiseleva1@dk3n55 ~/work/arch-pc/lab06 $

```

Рис. 4.22: Запуск исполняемого файла

Листинг 4.1. Программа для вычисления значения выражения $(10x -$

5)^2.

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной 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
mov ebx, 10 ; EBX=10
mul ebx ; EAX=EAX*10
add eax, -5 ; EAX=EAX*10-5
mul eax ; EAX=EAX*EAX
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 iprintLF ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

5 Выводы

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

6 Список литературы

1. Лабораторная работа №6