

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Беспутин Глеб Антонович

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	16
4.2.1	Ответы на вопросы по программе	19
5	Выводы	21
6	Список литературы	22

Список иллюстраций

4.1	Создание директории	9
4.2	Создание файла	9
4.3	Создание копии файла	10
4.4	Редактирование файла	11
4.5	Запуск исполняемого файла	12
4.6	Редактирование файла	12
4.7	Запуск исполняемого файла	13
4.8	Создание файла	13
4.9	Редактирование файла	14
4.10	Запуск исполняемого файла	14
4.11	Редактирование файла	15
4.12	Запуск исполняемого файла	15
4.13	Создание файла	16
4.14	Редактирование файла	16
4.15	Запуск исполняемого файла	16
4.16	Изменение программы	17
4.17	Запуск исполняемого файла	17
4.18	Создание файла	18
4.19	Редактирование файла	18
4.20	Запуск исполняемого файла	19

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. - 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 №7 (рис. [4.1]). Перехожу в созданный каталог с помощью утилиты `cd`.

```
[glebbesputin@fedora ~]$ mkdir ~/work/arch-pc/lab06  
[glebbesputin@fedora ~]$ cd ~/work/arch-pc/lab06  
[glebbesputin@fedora lab06]$ touch lab6-1.asm  
[glebbesputin@fedora lab06]$
```

Рис. 4.1: Создание директории

С помощью утилиты `touch` создаю файл `lab7-1.asm` (рис. [4.2]).

```
[glebbesputin@fedora ~]$ mkdir ~/work/arch-pc/lab06  
[glebbesputin@fedora ~]$ cd ~/work/arch-pc/lab06  
[glebbesputin@fedora lab06]$ touch lab6-1.asm  
[glebbesputin@fedora lab06]$
```

Рис. 4.2: Создание файла

Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться в других программах (рис. [4.3]).

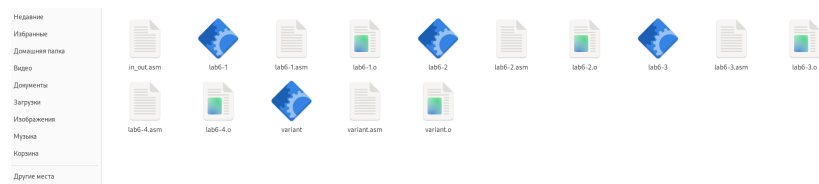


Рис. 4.3: Создание копии файла

Открываю созданный файл lab7-1.asm, вставляю в него программу вывода значения регистра eax (рис. [4.4]).

```
%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 sprintf
call quit
```

Рис. 4.4: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. [4.5]). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

```
[glebbesputin@fedora lab06]$ nasm -f elf lab6-1.asm
[glebbesputin@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[glebbesputin@fedora lab06]$ ./lab6-1
j
```

Рис. 4.5: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. [4.6]).

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

Рис. 4.6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. [4.7]). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

```
[glebbesputin@fedora lab06]$ nasm -f elf lab6-1.asm
[glebbesputin@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[glebbesputin@fedora lab06]$ ./lab6-1
```

Рис. 4.7: Запуск исполняемого файла

Создаю новый файл lab7-2.asm с помощью утилиты touch (рис. [4.8]).

```
[glebbesputin@fedora lab06]$ nasm -f elf lab6-2.asm
[glebbesputin@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[glebbesputin@fedora lab06]$ ./lab6-2
106
```

Рис. 4.8: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (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 исполняемый файл lab7-2 (рис. [4.10]). 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”.

```

[glebbesputin@fedora lab06]$ nasm -f elf lab6-2.asm
[glebbesputin@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[glebbesputin@fedora lab06]$ ./lab6-2
106

```

Рис. 4.10: Запуск исполняемого файла

Заменяю в тексте программы в файле lab7-2.asm символы “6” и “4” на числа 6 и 4 (рис. [4.11]).

```

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

```

Рис. 4.11: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. [4.12]).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```

[glebbesputin@fedora lab06]$ nasm -f elf lab6-2.asm
[glebbesputin@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[glebbesputin@fedora lab06]$ ./lab6-2
10

```

Рис. 4.12: Запуск исполняемого файла

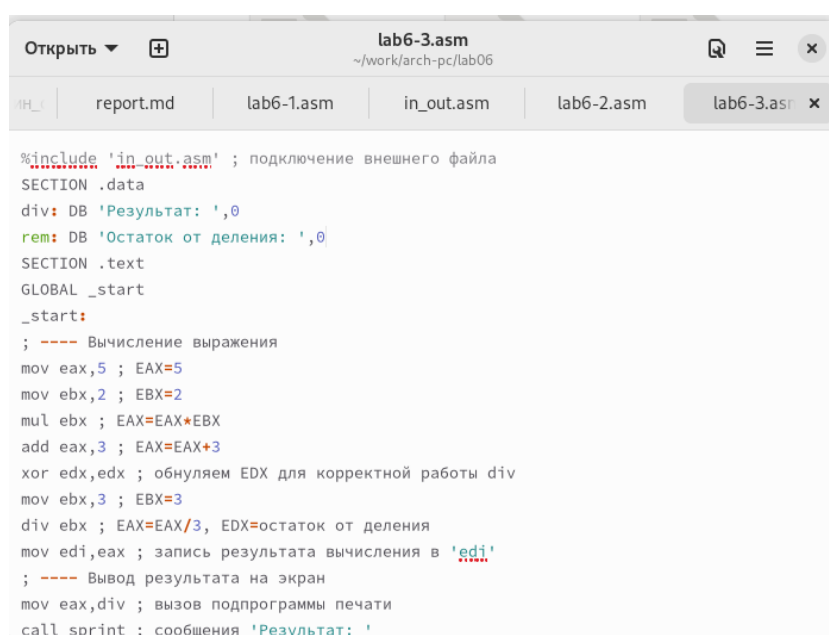
4.2 Выполнение арифметических операций в NASM

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

```
[glebbesputin@fedora lab06]$ touch ~/work/arch-pc/lab06/lab6-3.asm
[glebbesputin@fedora lab06]$
```

Рис. 4.13: Создание файла

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



```
%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: '
```

Рис. 4.14: Редактирование файла

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

```
[glebbesputin@fedora lab06]$ nasm -f elf lab6-3.asm
[glebbesputin@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[glebbesputin@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рис. 4.15: Запуск исполняемого файла

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

```
; ---- 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
```

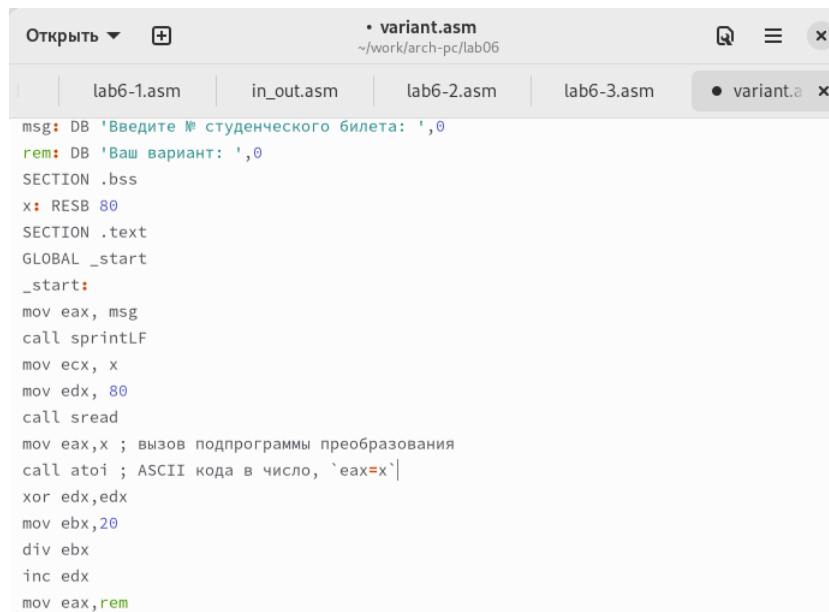
Рис. 4.16: Изменение программы

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

```
[glebbesputin@fedora lab06]$ nasm -f elf lab6-3.asm
[glebbesputin@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[glebbesputin@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рис. 4.17: Запуск исполняемого файла

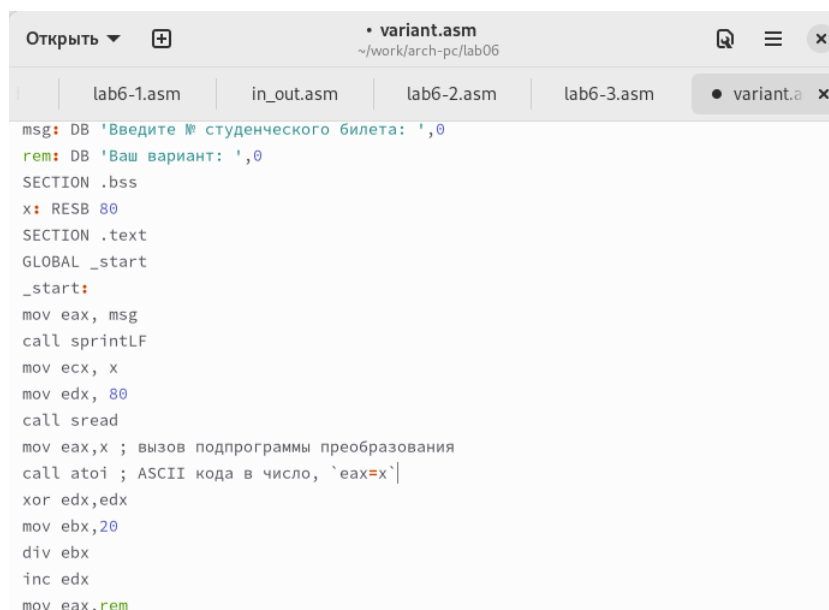
Создаю файл variant.asm с помощью утилиты touch (рис. [4.18]).



```
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
```

Рис. 4.18: Создание файла

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

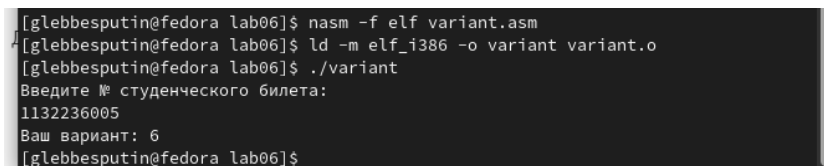


```
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
```

Рис. 4.19: Редактирование файла

Создаю и запускаю исполняемый файл (рис. [4.20]). Ввожу номер своего студ.

билета с клавиатуры, программа вывел, что мой вариант - .6



```
[glebbesputin@fedora lab06]$ nasm -f elf variant.asm
[glebbesputin@fedora lab06]$ ld -m elf_i386 -o variant variant.o
[glebbesputin@fedora lab06]$ ./variant
Введите № студенческого билета:
1132236005
Ваш вариант: 6
[glebbesputin@fedora 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
```

5 Выводы

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

6 Список литературы

1. Лабораторная работа №7
2. Таблица ASCII