

Лабораторная работа №6

Дисциплина: Архитектура компьютера

Карпачев Ярослав Олегович

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	13
4.2.1	Ответы на вопросы по программе	16
4.3	Выполнение заданий для самостоятельной работы	17
5	Выводы	19

Список иллюстраций

4.1	Создание файла lab6-1.asm	8
4.2	Редактированный файл	9
4.3	Создание исполняемого кода и его запуск	9
4.4	Редактированный файла	10
4.5	Создание исполняемого файла и его запуск	10
4.6	Создание файла lab6-2.asm	11
4.7	Редактированный файла	11
4.8	Компиляция файла и его запуск	12
4.9	Редактирование программы	12
4.10	запуск файла	13
4.11	запуск нового варианта	13
4.12	Создание файла	13
4.13	Редактирование файла	14
4.14	Запуск исполняемого файла	14
4.15	Запуск нового исполняемого файла	15
4.16	Создание файла	15
4.17	Редактирование файла	15
4.18	Запуск файла	16
4.19	Код программы	17
4.20	Запуск программы для выполнения задания для самостоятельной работы	18

1 Цель 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.

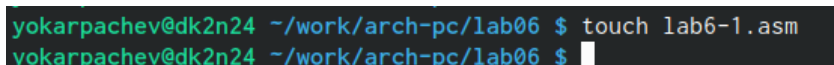
Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно 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 вывести 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` создал каталог для программ лабораторной работы № 6, перешел в него и создал файл `lab6-1.asm`. Скопировал в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться. (рис. 4.1).



```
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ touch lab6-1.asm
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $
```

Рис. 4.1: Создание файла `lab6-1.asm`

Ввел в файл `lab6-1.asm` текст программы из листинга 6.1 (рис. 4.2).


```

#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. 4.2: Редактированный файл

Создал исполняемый файл, скомпилировал и запустил его (рис. 4.3). Вывод - символ j, потому что программа вывела символ по системе ASCII сумме двоичных кодов символов 4 и 6.

```

yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ./lab6-1
j
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $

```

Рис. 4.3: Создание исполняемого кода и его запуск

Исправил текст программы: поменяла символы “6” и “4” на цифры 6 и 4 (рис. 4.4). Скомпилировал файл и запустил его (рис. 4.5). Теперь вывелся символ с кодом 10, это символ перевода строки, поэтому он не отображается при выводе на экран.

```

#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: Редактированный файла

```

yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ vim lab6-1.asm
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ./lab6-1

yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ █

```

Рис. 4.5: Создание исполняемого файла и его запуск

Создал lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввел текст программы из листинга 6.2 (рис. 4.6) и (рис. 4.7).

```
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ touch lab6-2.asm
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $
```

Рис. 4.6: Создание файла lab6-2.asm

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

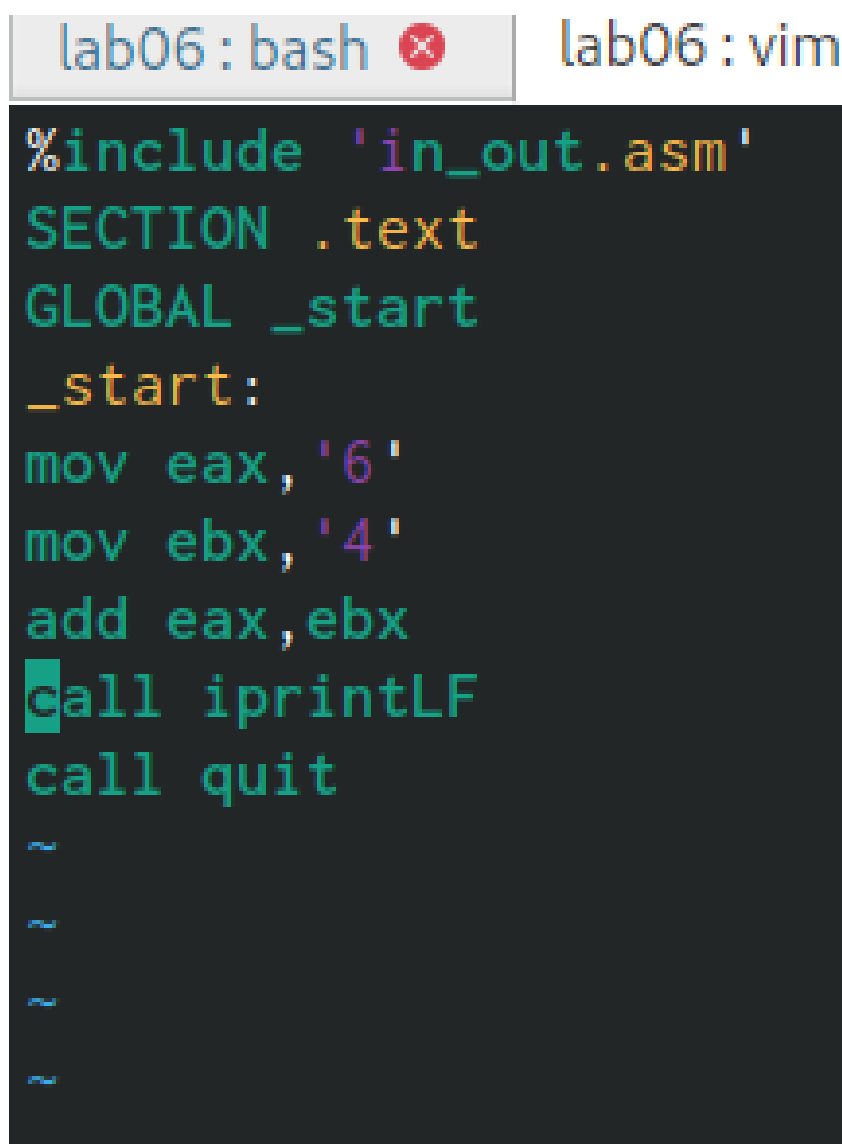
Рис. 4.7: Редактированный файла

Скомпилировал файл и запустил его (рис. 4.8). Теперь вывело число 106, потому что программа позволяет вывести число, а не символ, хотя происходит сложение кодов символов “6” и “4”.

```
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ nasm -felf lab6-2.asm
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ./lab6-2
106
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $
```

Рис. 4.8: Компиляция файла и его запуск

Заменял в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.9).



```
lab06 : bash ✖ lab06 : vim
%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 программы

Создал и запустил новый исполняемый файл (рис. 4.10). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ nasm -felf lab6-2.asm
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ./lab6-2
10
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $
```

Рис. 4.10: запуск файла

Заменяю в тексте программы `iprintLF` на `iprint` (рис. 4.11). Создал и скомпилировал файл (рис. 4.11). Вывод не изменился - 10, но теперь `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF`.

```
10
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ nasm -felf lab6-2.asm
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ./lab6-2
10yokarpachev@dk2n24 ~/work/arch-pc/lab06 $
```

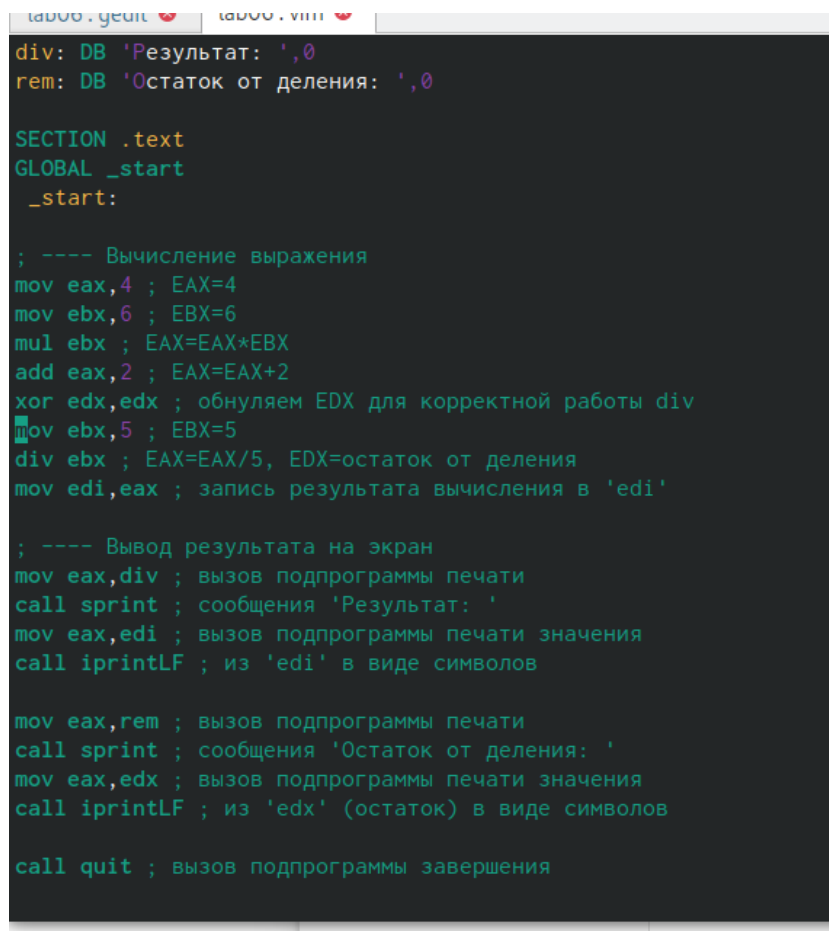
Рис. 4.11: запуск нового варианта

4.2 Выполнение арифметических операций в NASM

Создал файл `lab6-3.asm` с помощью утилиты `touch` (рис. 4.12). Ввел в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 4.12) и (рис. 4.13).

```
in_04c.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ touch lab6-3.asm
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $
```

Рис. 4.12: Создание файла



```
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'

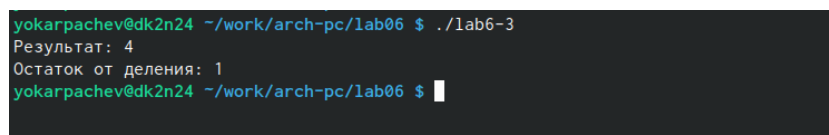
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

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
```

Рис. 4.13: Редактирование файла

Скомпилировал файл и запустил его (рис. [4.14]).



```
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $
```

Рис. 4.14: Запуск исполняемого файла

Изменил программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$. Создал и запустил новый исполняемый файл (рис. 4.15). Программа отработала верно.

```
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $
```

Рис. 4.15: Запуск нового исполняемого файла

Создал файл `variant.asm` с помощью `touch` (рис. 4.16). Ввел код из листинга 6.4 (рис. 4.17). Создала исполняемый файл и запустила его. Ввела номер своего студенческого билета и получила номер своего варианта – 11 (рис. 4.18). Проверила правильность выполнения, вычислив номер варианта аналитически.

```
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ touch variant.asm
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $
```

Рис. 4.16: Создание файла

```

;-----
; Программа вычисления варианта
;-----
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите No 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 iprintLF
"variant.asm" 31L, 621B

```

Рис. 4.17: Редактирование файла

```

yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132232862
Ваш вариант: 3
yokarpachev@dk2n24 ~/work/arch-pc/lab06 $

```

Рис. 4.18: Запуск 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

Создал myprogram.asm. Открыл его, ввела код программы для вычисления значения выражения $(x + 2)^2$ (вариант 3) (рис. 4.19). Создал и запустил исполняемый файл. При вводе значения 1 на входе вывод программы = 10. При вводе значения 7 на входе вывод программы = 70. Значит, программа работает верно. (рис. 4.20).

```
1 %include 'in_out.asm'  
2 SECTION .data  
3 msg: DB 'Введите x: ',0  
4 rem: DB 'Ответ: ',0  
5 SECTION .bss  
6 x: RESB 80  
7 SECTION .text  
8 GLOBAL _start  
9 _start:  
10 mov eax, msg  
11 call sprintLF  
12 mov ecx, x  
13 mov edx, 80  
14 call sread  
15 mov eax, x ;  
16 call atoi ;  
17 add eax, 2  
18 mov ebx, eax  
19 mul ebx  
20 mov ebx, eax  
21 mov eax, rem  
22 call sprint  
23 mov eax, ebx  
24 call iprintLF  
25 call quit
```

Рис. 4.19: Код программы

```
yokarpachev@dk8n75 ~/work/arch-pc/lab06 $ ./myprogram
Введите x:
2
Ответ: 16yokarpachev@dk8n75 ~/work/arch-pc/lab06 $ ./myprogram
Введите x:
8
Ответ: 100yokarpachev@dk8n75 ~/work/arch-pc/lab06 $
```

Рис. 4.20: Запуск программы для выполнения задания для самостоятельной работы

5 Выводы

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