

Отчёт по лабораторной работе №7

Арифметические операции в NASM

Кочина Дарья Сергеевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Выводы	25

Список иллюстраций

4.1	Создание файла lab7-1.asm	7
4.2	Текст в файле lab7-1.asm	8
4.3	Создание файла и проверка работы	8
4.4	Изменения в программе lab6-1.asm	10
4.5	Проверка программы	11
4.6	Создание файла lab7-2.asm	11
4.7	Текст программы из листинга	12
4.8	Запуск программы lab7-2.asm	12
4.9	Изменения в программе lab6-2.asm	13
4.10	Проверка программы	13
4.11	Замена <code>iprintLF</code> на <code>iprint</code>	14
4.12	Запуск программы с изменениями	14
4.13	Создание файла lab7-3.asm	15
4.14	Текст программы из листинга	15
4.15	Запуск программы lab7-3.asm	16
4.16	Изменения в программе lab7-3.asm	17
4.17	Запуск программы	17
4.18	Создание файла <code>variant.asm</code>	18
4.19	Текст программы из листинга	19
4.20	Запуск программы <code>variant.asm</code>	20
4.21	Создание файла lab7-4.asm	21
4.22	Программа для вычисления в файле lab7-4.asm	22
4.23	Программа для вычисления в файле lab7-4.asm	23
4.24	Результаты работы программы	24

1 Цель работы

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

2 Задание

Освоить арифметические инструкции языка ассемблера NASM.

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. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. 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.

4 Выполнение лабораторной работы

1. Я создала каталог для программ лабораторной работы № 7, перешла в него и создала файл lab7-1.asm. (рис. 4.1)

```
dskochina@dk5n60 ~ $ mkdir ~/work/study/2022-2023/Архитектура\ компьютера/arch-pc/lab07
dskochina@dk5n60 ~ $ cd ~/work/study/2022-2023/Архитектура\ компьютера/arch-pc/lab07
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ pwd
/afs/.dk.sci.pfu.edu.ru/home/d/s/dskochina/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ls
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ touch lab7-1.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ls
lab7-1.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $
```

Рис. 4.1: Создание файла lab7-1.asm

2. Я ввела нужный текст в файле lab7-1.asm, создала файл и проверила его. Программа вывела “j”. (рис. 4.2, 4.3)

```

#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: Текст в файле lab7-1.asm

```

dskochina@dk5n60 ~ $ cd ~/work/study/2022-2023/Архитектура\ компьютера/arch-pc/lab07
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf lab7-1.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ./lab7-1
j
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ 

```

Рис. 4.3: Создание файла и проверка работы

3. Я изменила текст программы и вместо символов записала в регистры числа. Я создала исполняемый файл и запустила его. Программа вывела невидимый символ. (рис. 4.4, 4.5)

```
%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: Изменения в программе lab6-1.asm

```

dskochina@dk5n60 ~ $ cd ~/work/study/2022-2023/Архитектура\ компьютера/arch-pc/lab07
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf lab7-1.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ./lab7-1

dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ./lab7-1

dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ █

```

Рис. 4.5: Проверка программы

4. Я создала файл lab7-2.asm и ввела в него текст программы. (рис. 4.6, 4.7)

```

dskochina@dk5n60 ~ $ cd ~/work/study/2022-2023/Архитектура\ компьютера/arch-pc/lab07
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ touch lab7-2.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ █

```

Рис. 4.6: Создание файла lab7-2.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 из листинга

5. Я создала исполняемый файл и запустила его. В результате работы программы я получила число 106. (рис. 4.8)

```

dskochina@dk5n60 ~ $ cd ~/work/study/2022-2023/Архитектура\ компьютера/arch-pc/lab07
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf lab7-2.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ./lab7-2
106
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ 

```

Рис. 4.8: Запуск программы lab7-2.asm

6. Я изменила текст программы lab7-2.asm и запустила её. Она вывела число 10. (рис. 4.9, 4.10)

```

#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: Изменения в программе lab6-2.asm

```

dskochina@dk5n60 ~ $ cd ~/work/study/2022-2023/Архитектура\ компьютера/arch-pc/lab07
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf lab7-2.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ./lab7-2
10
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ 

```

Рис. 4.10: Проверка программы

7. Я заменила функцию iprintLF на iprint. Создала исполняемый файл и запустила его. Отличие вывода функций iprintLF от iprint заключается в том, что при использовании команды iprintLF мы начинаем вводить команду на следующей строке, а при использовании команды iprint мы вводим следующие данные на той же строке. (рис. 4.11, 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: Замена iprintLF на iprint

```
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf lab7-2.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ./lab7-2
10dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $
```

Рис. 4.12: Запуск программы с изменениями

8. Я создала файл lab7-3.asm и ввела текст программы из листинга. Создала исполняемый файл и запустила его. (рис. 4.13, 4.14, 4.15)

```

10dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ touch lab7-3.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.o lab7-3.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ 

```

Рис. 4.13: Создание файла lab7-3.asm

```

#include 'in_out.asm' ; подключение внешнего файла

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

SECTION .text
GLOBAL _start
_start:

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'

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. 4.14: Текст программы из листинга

```
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf lab7-3.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ./lab7-3
Результат: 4
Остаток от деления: 1
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $
```

Рис. 4.15: Запуск программы lab7-3.asm

9. Я изменила текст программы для вычисления выражения $f(x)=(4*6+2)/5$.
Создала исполняемый файл и проверила его работу. (рис. 4.16, 4.17)


```

#include 'in_out.asm' ; подключение внешнего файла

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

SECTION .text
GLOBAL _start
_start:

mov eax,4
mov ebx,6
mul ebx

add eax,2
xor edx,edx
mov ebx,5
div ebx

mov edi,eax

mov eax,div
call sprint
mov eax,edi
call iprintLF

mov eax,rem
call sprint
mov eax,edx
call iprintLF

call quit

```

Рис. 4.16: Изменения в программе lab7-3.asm

```

dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf lab7-3.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ./lab7-3
Результат: 5
Остаток от деления: 1
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ 

```

Рис. 4.17: Запуск программы

10. Я создала файл variant.asm. Ввела текст программы из листинга в файл. (рис.

4.18, 4.19)

```
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ touch variant.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.o lab7-3 lab7-3.asm lab7-3.o variant.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $
```

Рис. 4.18: Создание файла variant.asm

```

#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 sprintfLF

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

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

mov eax, rem
call sprintf
mov eax, edx
call iprintfLF

call quit

```

Рис. 4.19: Текст программы из листинга

11. Я создала исполняемый файл и запустила его. Проверила работу программы,

вычислив номер варианта аналитически. Результат совпал. (рис. 4.20)

```
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf variant.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o variant variant.o
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ./variant
Введите No студенческого билета:
1132226493
Ваш вариант: 14
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $
```

Рис. 4.20: Запуск программы variant.asm

Вопросы

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

Строки `mov eax,rem` `call sprint`.

2. Для чего используются следующие инструкции `nasm`: `mov ecx,x`, `mov edx,80`, `call sread`?

Инструкция `mov ecx,x` записывает адреса выводимого сообщения в EAX. Инструкция `mov edx,80` записывает длину вводимого сообщения в EBX. Инструкция `call sread` выполняет вызов программы ввода сообщения.

3. Для чего используется инструкция “`call atoi`”?

Эта инструкция используется для преобразования кода переменной ASCII в число.

4. Какие строки листинга 7.4 отвечают за вычисления варианта?

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

5. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”?

Остаток от деления при выполнении инструкции “`div ebx`” записывается в регистр EBX.

6. Для чего используется инструкция “inc edx”?

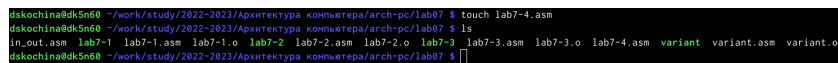
Эта инструкция используется для увеличения значения edx на единицу.

7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений?

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

Самостоятельная работа

1. Я создала файл lab7-4.asm для выполнения самостоятельной работы. (рис. 4.21)



```
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ touch lab7-4.asm
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.o lab7-3 lab7-3.asm lab7-3.o lab7-4.asm variant variant.asm variant.o
dskochina@dk5n60 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $
```

Рис. 4.21: Создание файла lab7-4.asm

2. Я написала программу вычисления выражения $f(x)=(x/2+8)3$. Программа вывела выражение для вычисления, вывела запрос на ввод значения x, вычислила заданное выражение в зависимости от введенного x, вывела результат вычислений. Вид функции $f(x)$ я выбрала из таблицы вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Я создала исполняемый файл и проверила его работу. Для этого я вводила значения для x, равные 1 и 4. Результаты работы программы были верными. При вводе 1 программа вывела ответ 24, т.к. учитывалась только целая часть от деления. При вводе 4 ответ оказался 30, что является верным. (рис. 4.22, 4.23, 4.24)

```

#include 'in_out.asm'

SECTION .data

prim: DB '(x/2+8)3',0
x1: DB 'Введите значение x: ',0
otv1: DB 'Ответ:',0

SECTION .bss
p: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,prim
call sprintLF

mov eax,x1
call sprint

mov ecx,p
mov edx,80
call sread

mov eax,p
call atoi

xor edx,edx

mov ebx,2
div ebx
add eax,8
xor ebx,ebx

```

Рис. 4.22: Программа для вычисления в файле lab7-4.asm

```
xor ebx,ebx
mov ebx,3
mul ebx

mov edi,eax

mov eax,otv1
call sprintLF
mov eax,edi
call iprintLF

call quit
```

Рис. 4.23: Программа для вычисления в файле lab7-4.asm

```

dskochina@dk8n54 ~ $ cd ~/work/study/2022-2023/Архитектура\ компьютера/arch-pc/lab07
dskochina@dk8n54 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf lab7-4.asm
dskochina@dk8n54 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
dskochina@dk8n54 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ./lab7-4
(x/2+8)3
Введите значение x: 1
Ответ:
24
dskochina@dk8n54 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ ./lab7-4
(x/2+8)3
Введите значение x: 4
Ответ:
30
dskochina@dk8n54 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07 $ █

```

Рис. 4.24: Результаты работы программы

5 Выводы

В ходе выполнения данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.