

Лабораторная работа №7

Арифметические операции в NASM.

Воробчук Лилия Андреевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Вопросы	12
4	Самостоятельная работа	13
5	Выводы	16

Список иллюстраций

2.1	создание каталога	6
2.2	ввод листинга программы	6
2.3	копирование файла	6
2.4	исполнение программы	7
2.5	новая программа	7
2.6	исполнение программы	7
2.7	создание файла	7
2.8	ввод листинга программы	8
2.9	исполнение программы	8
2.10	исполнение программы	8
2.11	запуск измененной программы	8
2.12	создание файла	9
2.13	ввод листинга программы	9
2.14	исполнение программы	9
2.15	измененная программа	10
2.16	исполнение программы	10
2.17	ввод листинга программы	11
2.18	вычисление номера варианта	11
4.1	программа	14
4.2	вычисление	15

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

1. Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рис. 2.1)

```
lavorobchuk@dk1n22 ~ $ mkdir ~/work/arch-pc/lab07
lavorobchuk@dk1n22 ~ $ cd ~/work/arch-pc/lab07
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 2.1: создание каталога

2. Ввожу в файл lab7-1.asm текст программы из листинга (рис. 2.2)



```
lab7-1.asm
~/work/arch-pc/lab07

1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1: RESB 80
5
6 SECTION .text
7 GLOBAL _start
8
9 _start:
10
11 mov eax, '6'
12 mov ebx, '4'
13 add eax, ebx
14 mov [buf1], eax
15 mov eax, buf1
16 call sprintLF
17
18 call quit
```

Рис. 2.2: ввод листинга программы

3. Для корректной работы программы подключаемый файл in_out.asm копирую в каталог ~/work/arch-pc/lab07. (рис. 2.3)



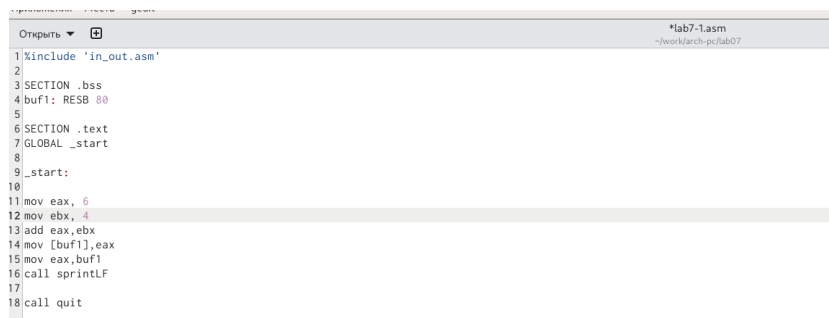
Рис. 2.3: копирование файла

4. Создаю исполняемый файл и запускаю его (рис. 2.4)

```
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-1
j
```

Рис. 2.4: исполнение программы

5. Меняю текст программы и вместо символов записываю в регистры числа.
Для этого нужно убрать кавычки (рис. 2.5)



```
1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1: RESB 80
5
6 SECTION .text
7 GLOBAL _start
8
9 _start:
10
11 mov eax, 6
12 mov ebx, 4
13 add eax, ebx
14 mov [buf1], eax
15 mov eax, buf1
16 call sprintf
17
18 call quit
```

Рис. 2.5: новая программа

6. Создаю исполняемый файл и запускаю его (рис. 2.6)

```
j
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-1
```

Рис. 2.6: исполнение программы

7. Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 (рис. 2.7)

```
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ touch lab7-2.asm
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $
```

Рис. 2.7: создание файла

8. Ввожу в него текст программы из листинга 7.2.(рис. 2.8)

```

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. 2.8: ввод листинга программы

9. Создаю исполняемый файл и запускаю его (рис. 2.9)

```

lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-2
106

```

Рис. 2.9: исполнение программы

10. Аналогично предыдущему примеру заменяю символы на числа и запускаю программу (рис. 2.10)

```

lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-2
10

```

Рис. 2.10: исполнение программы

11. Заменяю функцию iprintLF на iprint. Создаю исполняемый файл и запускаю его. (рис. 2.11)

```

lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-2
10lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ █

```

Рис. 2.11: запуск измененной программы

Разница между функциями в том, что `iprint` просто выводит сообщение на экран, а `iprintLF` добавляет к этому переход на новую строку.

12. Создаю файл `lab7-3.asm` в каталоге `~/work/arch-pc/lab07` (рис. 2.12)

```
10lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ touch ~/work/arch-pc/lab07/lab7-3.asm
```

Рис. 2.12: создание файла

13. Изучаю текст программы из листинга 7.3 и ввожу в `lab7-3.asm`. (рис. 2.13)

```
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
10 _start:
11
12 mov eax,5
13 mov ebx,2
14 mul ebx
15
16 add eax,3
17 xor edx,edx
18 mov ebx,3
19 div ebx
20 mov edi,eax
21 mov eax,div
22 call sprint
23 mov eax,edi
24 call iprintLF
25
26 mov eax,rem
27 call sprint
28 mov eax,edx
29 call iprintLF
30
31 call quit
```

Рис. 2.13: ввод листинга программы

14. Создаю исполняемый файл и запускаю его (рис. 2.14)

```
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-3
Результат: 4
Остаток от деления: 1
```

Рис. 2.14: исполнение программы

15. Изменяю текст программы для вычисления выражения $\boxtimes(\boxtimes) = (4 \boxtimes 6 + 2)/5$ (рис. 2.15)

```

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
10 _start:
11
12 mov eax,4
13 mov ebx,6
14 mul ebx
15
16 add eax,2
17 xor edx,edx
18 mov ebx,5
19 div ebx
20 mov edi,eax
21 mov eax,div
22 call sprint
23 mov eax,edi
24 call iprintLF
25
26 mov eax,rem
27 call sprint
28 mov eax,edx
29 call iprintLF
30
31 call quit

```

Рис. 2.15: измененная программа

16. Создаю исполняемый файл и запускаю его (рис. 2.16)

```

lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-3
Результат: 5
Остаток от деления: 1

```

Рис. 2.16: исполнение программы

17. Создаю файл variant.asm в каталоге ~/work/arch-pc/lab07 и ввожу в этот файл текст программы из листинга 7.4 (рис. 2.17)

```

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 SECTION .text
10 GLOBAL _start
11 _start:
12
13 mov eax, msg
14 call sprintf
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax, x
19 call atoi
20 xor edx,edx
21 mov ebx, 20
22 div ebx
23 inc edx
24 mov eax,rem
25 call sprintf
26 mov eax,edx
27 call iprintLF
28 call quit

```

Рис. 2.17: ввод листинга программы

18. Создаю исполняемый файл, запускаю его и вывожу номер варианта на экран (рис. 2.18)

```

Новая вкладка  Разделить окно
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf variant.asm
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o variant variant.o
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $ ./variant
Введите № студенческого билета:
1132226486
Ваш вариант: 7
lavorobchuk@dk1n22 ~/work/arch-pc/lab07 $

```

Рис. 2.18: вычисление номера варианта

3 Вопросы

1) Какие строки листинга 7.4 отвечают за вывод на экран сообщения 'Ваш вариант: '? `mov eax,msg call sprintLF` 2) Для чего используются следующие инструкции? `mov ecx,x mov edx,80 call sread` Эти инструкции используются для ввода переменной X с клавиатуры и сохранения введенных данных. 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? `mov ebx,20 div ebx inc edx` 5) В какой регистр записывается остаток от деления при выполнении инструкции "div ebx"? В регистре ebx. 6) Для чего используется инструкция "inc edx"? Для увеличения значения edx на 1. 7) Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений? `mov eax,edx call iprintLF`

4 Самостоятельная работа

В результате работы программы `variant.asm` получаю вариант №7. Создаю файл с именем `sus.asm`, пишу там программу для вычисления значения выражения $5(x-1)^2$ (рис. 4.1)

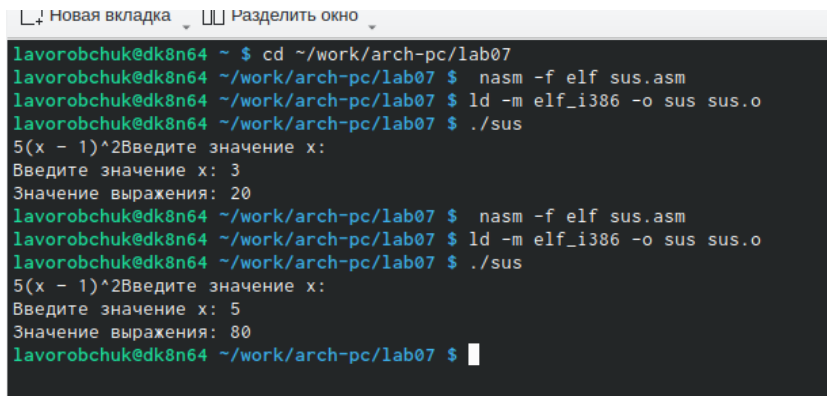
```

1 %include 'in_out.asm'
2
3 SECTION .data
4 prim: DB '5(x - 1)^2'
5 msg: DB 'Введите значение x: ',0
6 rem: DB 'Значение выражения: ',0
7
8 SECTION .bss
9 p: RESB 80
10
11 SECTION .text
12 GLOBAL _start
13 _start:
14
15 mov eax,prim
16 call sprintLF
17
18 mov eax,msg
19 call sprint
20
21 mov ecx,p
22 mov edx,80
23 call sread
24
25 mov eax,p
26 call atoi
27
28 dec eax
29 mov ebx,eax
30 mul ebx
31 mov ecx,5
32 mul ecx
33
34 mov edi,eax
35
36 mov eax,rem
37 call sprint
38 mov eax,edi
39 call iprintLF
40
41 call quit

```

Рис. 4.1: программа

Далее запускаю файл. Подставляю заданные значения x , сверяю полученные результаты с вычисленными аналитически. (рис. 4.2)



```
lavorobchuk@dk8n64 ~ $ cd ~/work/arch-pc/lab07
lavorobchuk@dk8n64 ~/work/arch-pc/lab07 $ nasm -f elf sus.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o sus sus.o
lavorobchuk@dk8n64 ~/work/arch-pc/lab07 $ ./sus
5(x - 1)^2Введите значение x:
Введите значение x: 3
Значение выражения: 20
lavorobchuk@dk8n64 ~/work/arch-pc/lab07 $ nasm -f elf sus.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o sus sus.o
lavorobchuk@dk8n64 ~/work/arch-pc/lab07 $ ./sus
5(x - 1)^2Введите значение x:
Введите значение x: 5
Значение выражения: 80
lavorobchuk@dk8n64 ~/work/arch-pc/lab07 $
```

Рис. 4.2: вычисление

5 Выводы

В ходе выполнения данной лабораторной работы мной были освоены арифметические инструкции языка ассемблера NASM