

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Батова Ирина Сергеевна, НММбд-01-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Ответы на вопросы	13
4	Задание для самостоятельной работы	15
5	Выводы	17

Список иллюстраций

2.1	Создание необходимых для работы каталогов и файлов	6
2.2	Ввод программы листинга 7.1	6
2.3	Запуск программы lab7-1	7
2.4	Внесение изменений в файл с программой lab7-1	7
2.5	Запуск измененной программы lab7-1	7
2.6	Создание файла 'lab7-2.asm'	8
2.7	Ввод программы листинга 7.2	8
2.8	Запуск программы lab7-2	8
2.9	Внесение изменений в файл с программой lab7-2	9
2.10	Запуск измененной программы lab7-2	9
2.11	Замена команды irpintLF на iprint	9
2.12	Запуск дважды измененной программы lab7-2	10
2.13	Создание файла 'lab7-3.asm'	10
2.14	Ввод программы листинга 7.3	10
2.15	Запуск программы lab7-3	11
2.16	Внесение изменений в файл с программой lab7-3	11
2.17	Запуск измененной программы lab7-3	11
2.18	Создание файла 'variant.asm'	12
2.19	Ввод программы листинга 7.4	12
2.20	Запуск программы variant	12
4.1	Ввод программы для вычисления значений функции	15
4.2	Запуск программы sam	16

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

1. Сначала создаем каталог для программ данной лабораторной работы, используя команду 'mkdir'. Далее переходим в этот каталог с помощью команды 'cd' и создаем файл 'lab7-1.asm' (рис. 2.1).

```
[isbatova@fedora ~]$ mkdir ~/work/arch-pc/lab07
[isbatova@fedora ~]$ cd ~/work/arch-pc/lab07
[isbatova@fedora lab07]$ touch lab7-1.asm
[isbatova@fedora lab07]$
```

Рис. 2.1: Создание необходимых для работы каталогов и файлов

2. Открываем файл 'lab7-1.asm' и вводим в него программу из листинга 7.1 из лабораторной работы (рис. 2.2).



Рис. 2.2: Ввод программы листинга 7.1

Перед запуском файла копируем файл 'in_out.asm' из каталога lab06 в каталог lab07 для корректной работы программы.

После этого создаем исполняемый файл и запускаем его (рис. 2.3). Наша программа выводит 'j' - это происходит потому, что программа складывает код символа 'б' и код символа '4', и выводит символ, соответствующий сумме этих кодов.

```
[isbatova@fedora lab07]$ nasm -f elf lab7-1.asm
[isbatova@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[isbatova@fedora lab07]$ ./lab7-1
j
[isbatova@fedora lab07]$
```

Рис. 2.3: Запуск программы lab7-1

3. Открываем файл 'lab7-1.asm' и немного видоизменяем программу: вместо символов 'б' и '4' записываем числа 6 и 4 соответственно (рис. 2.4).

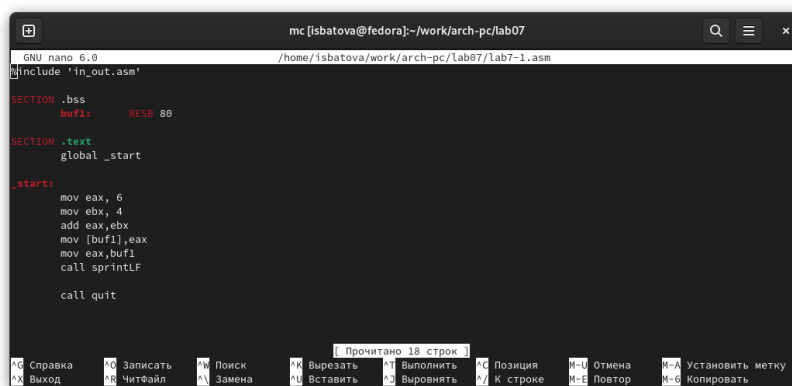


Рис. 2.4: Внесение изменений в файл с программой lab7-1

Создаем исполняемый файл и запускаем его (рис. 2.5). В данном случае у нас будет выводиться символ с кодом 10. По таблице определяем, что это символ перевода строки, поэтому программа ничего не выводит, а только переходит на следующую строку.

```
[isbatova@fedora lab07]$ nasm -f elf lab7-1.asm
[isbatova@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[isbatova@fedora lab07]$ ./lab7-1

[isbatova@fedora lab07]$
```

Рис. 2.5: Запуск измененной программы lab7-1

4. В том же каталоге создаем файл 'lab7-2.asm', используя команду 'touch' (рис. 2.6).

```
[isbatova@fedora lab07]$ touch lab7-2.asm  
[isbatova@fedora lab07]$
```

Рис. 2.6: Создание файла 'lab7-2.asm'

Далее открываем этот файл и вводим в него программу из листинга 7.2 из лабораторной работы (рис. 2.7).



Рис. 2.7: Ввод программы листинга 7.2

Создаем исполняемый файл и запускаем его (рис. 2.8). Программа выводит нам число 106, так как аналогично складываются коды символов '6' и '4', но благодаря функции iprintLF выводится само число (а не символ, которому соответствует данное число).

```
[isbatova@fedora lab07]$ nasm -f elf lab7-2.asm  
[isbatova@fedora lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2  
[isbatova@fedora lab07]$ ./lab7-2  
106  
[isbatova@fedora lab07]$
```

Рис. 2.8: Запуск программы lab7-2

5. Далее открываем файл 'lab7-2.asm' и видоизменяем его также, как изменяли первый файл: вместо символов '6' и '4' записываем числа 6 и 4 соответственно (рис. 2.9).



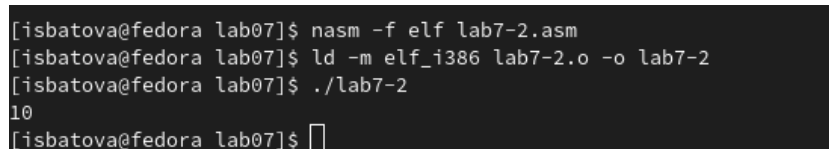
```
GNU nano 6.0 /home/isbatova/work/arch-pc/lab07/lab7-2.asm
#include 'in_out.asm'

SECTION .text
global _start

_start:
    mov eax, 6
    mov ebx, 4
    add eax, ebx
    call iprintLF
    call quit
```

Рис. 2.9: Внесение изменений в файл с программой lab7-2

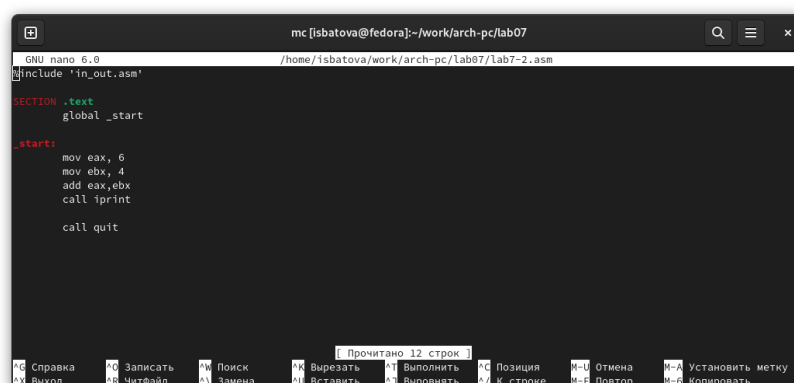
Создаем исполняемый файл и запускаем его (рис. 2.10). Наша программа выводит '10'.



```
[isbatova@fedora lab07]$ nasm -f elf lab7-2.asm
[isbatova@fedora lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2
[isbatova@fedora lab07]$ ./lab7-2
10
[isbatova@fedora lab07]$
```

Рис. 2.10: Запуск измененной программы lab7-2

Далее вновь открываем файл и вносим изменения - вместо iprintLF вводим iprint (рис. 2.11).



```
GNU nano 6.0 /home/isbatova/work/arch-pc/lab07/lab7-2.asm
#include 'in_out.asm'

SECTION .text
global _start

_start:
    mov eax, 6
    mov ebx, 4
    add eax, ebx
    call iprint
    call quit
```

Рис. 2.11: Замена команды iprintLF на iprint

Вновь создаем исполняемый файл и запускаем его (рис. 2.12). Наша программа

также выводит '10', но, в отличие от использования 'iprintLF', 'iprint' не делает переход на новую строку.

```
[isbatova@fedora lab07]$ nasm -f elf lab7-2.asm
[isbatova@fedora lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2
[isbatova@fedora lab07]$ ./lab7-2
10[isbatova@fedora lab07]$
```

Рис. 2.12: Запуск дважды измененной программы lab7-2

6. С помощью команды 'touch' в каталоге lab07 создаем файл 'lab7-3.asm' (рис. 2.13).

```
10[isbatova@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-3.asm
[isbatova@fedora lab07]$
```

Рис. 2.13: Создание файла 'lab7-3.asm'

Открываем его и вводим программу из листинга 7.3 из лабораторной работы (рис. 2.14).



Рис. 2.14: Ввод программы листинга 7.3

Создаем исполняемый файл и запускаем его (рис. 2.15). Видим, что программа работает корректно.

```
[isbatova@fedora lab07]$ nasm -f elf lab7-3.asm
[isbatova@fedora lab07]$ ld -m elf_i386 lab7-3.o -o lab7-3
[isbatova@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[isbatova@fedora lab07]$
```

Рис. 2.15: Запуск программы lab7-3

Далее вновь открываем этот файл и вносим изменения так, чтобы у нас вычислялось выражение $f(x) = (4 \cdot 6 + 2) / 5$ (рис. 2.16).

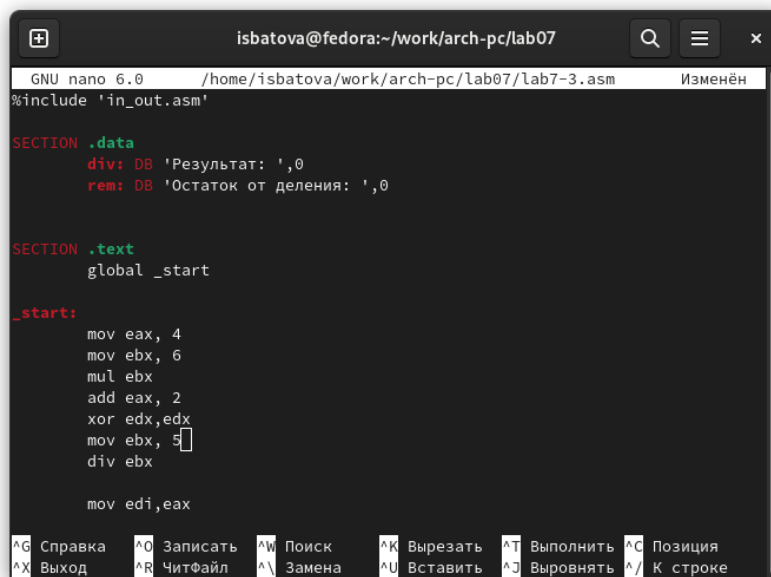


Рис. 2.16: Внесение изменений в файл с программой lab7-3

Создаем исполняемый файл и запускаем его (рис. 2.17). Программа работает корректно: действительно, если мы устно посчитаем уравнение, то получим 5 с остатком 1.

```
[isbatova@fedora lab07]$ nasm -f elf lab7-3.asm
[isbatova@fedora lab07]$ ld -m elf_i386 lab7-3.o -o lab7-3
[isbatova@fedora lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[isbatova@fedora lab07]$
```

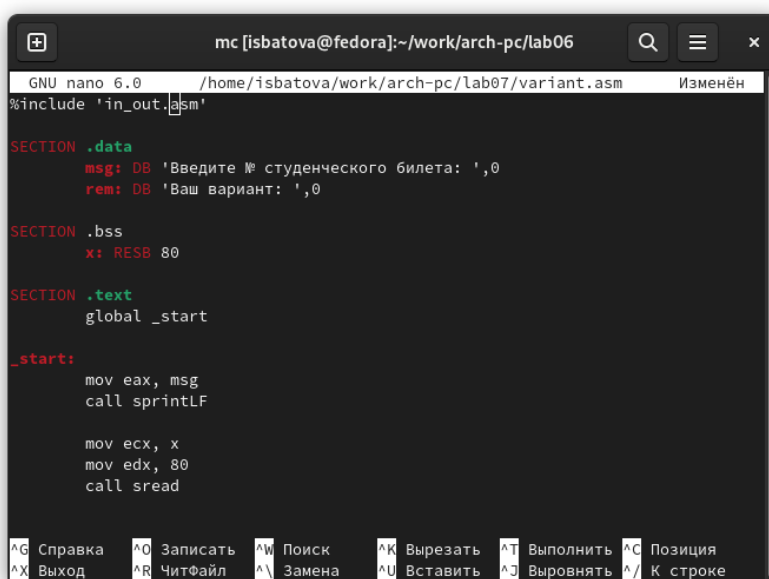
Рис. 2.17: Запуск измененной программы lab7-3

7. В каталоге lab07 создаем файл 'variant.asm', используя команду 'touch' (рис. 2.18).

```
[isbatova@fedora ~]$ touch ~/work/arch-pc/lab07/variant.asm
[isbatova@fedora ~]$
```

Рис. 2.18: Создание файла 'variant.asm'

Открываем данный файл, вводим в него программу из листинга 7.4 (рис. 2.19).



```
mc [isbatova@fedora]:~/work/arch-pc/lab06
GNU nano 6.0 /home/isbatova/work/arch-pc/lab07/variant.asm  Изменён
%include 'in_out.asm'

SECTION .data
    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 sprintLF

    mov ecx, x
    mov edx, 80
    call sread

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_/ К строке
```

Рис. 2.19: Ввод программы листинга 7.4

Создаем исполняемый файл и запускаем его (рис. 2.20). На предложение ввести номер студенческого билета вводим его и получаем свой вариант. Проверяем аналитически - вариант выведен верно.

```
[isbatova@fedora lab07]$ nasm -f elf variant.asm
[isbatova@fedora lab07]$ ld -m elf_i386 variant.o -o variant
[isbatova@fedora lab07]$ ./variant
Введите № студенческого билета:
1132226490
Ваш вариант: 11
[isbatova@fedora lab07]$
```

Рис. 2.20: Запуск программы variant

3 Ответы на вопросы

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

За это отвечают строки:

```
mov eax,rem call sprint
```

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

Данные инструкции используются для ввода сообщения (в данном случае `x`) и запись введенных данных в `ecx`.

3. Для чего используется инструкция `call atoi`?

Данная инструкция используется для преобразования символьного вида в число.

4. Какие строки листинга 7.4 отвечают за вычисления варианта?

За вычисление варианта отвечают строки:

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

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

Остаток от деления записывается в регистр `edx`.

6. Для чего используется инструкция “inc edx”?

Эта инструкция используется для прибавления к значению регистра edx единицу.

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

За вывод на экран результату вычислений отвечают строки:

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

4 Задание для самостоятельной работы

Создаем файл с именем 'sam.asm'. В последнем пункте лабораторной работы мне был выдан 11 вариант, поэтому программа была написана для функции $f(x)=10(x+1)-10$ (рис. 4.1).



```
GNU nano 2.9.3 /home/ysbatova/work/arch-pc/lab07/sam.asm
#include "io_out.asm"

SECTION .data
    primer DB "10(x+1)-10",0
    param DB "Введите значение x: ",0
    otvet DB "Ответ: ",0

SECTION .bss
    resb 80

SECTION .text
    global _start

_start:
    mov eax, primer
    call sprintf
    mov eax, param
    call sprintf
    mov ecx, x
    mov ebx, 80
    call read
    mov eax, x
    call atoi
    inc eax
    mov ebx, 10
    mul ebx
    sub eax, 10
    mov edi, eax
    mov eax, otvet
    call sprintf
    mov eax, edi
    call sprintf
    call quit
```

Рис. 4.1: Ввод программы для вычисления значений функции

Создаем исполняемый файл и запускаем его (рис. 4.2). На предложение ввести значение x вводим значения из таблицы: сначала 1, потом 7. При подсчете аналитически получаются такие же ответы, поэтому программа написана правильно.

```
[isbatova@fedora lab07]$ nasm -f elf sam.asm
[isbatova@fedora lab07]$ ld -m elf_i386 sam.o -o sam
[isbatova@fedora lab07]$ ./sam
10(x+1)-10
Введите значение x:
1
Ответ: 10
[isbatova@fedora lab07]$ ./sam
10(x+1)-10
Введите значение x:
7
Ответ: 70
```

Рис. 4.2: Запуск программы sam

5 Выводы

В данной лабораторной работе мной были освоены арифметические инструкции языка ассемблера NASM.