

Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Будник Александра Олеговна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выполнение заданий для самостоятельной работы	14
5	Выводы	16

Список иллюстраций

3.1	Создание директории	7
3.2	Создание файла	7
3.3	Код программы lab6-1	7
3.4	Запуск программы	8
3.5	Изменение кода	8
3.6	Запуск программы	8
3.7	Создание файла	9
3.8	Код программы lab6-2	9
3.9	Запуск программы	9
3.10	Изменение кода	10
3.11	Запуск программы	10
3.12	Изменение кода	10
3.13	Запуск программы	11
3.14	Создание файла	11
3.15	Код программы lab6-3	11
3.16	Запуск программы	12
3.17	Изменение кода	12
3.18	Запуск программы	12
3.19	Код программы variant	13
3.20	Запуск программы	13
4.1	Код программы lab6-4	14
4.2	Линковка	15
4.3	Запуск программы	15

Список таблиц

2.1	Описание некоторых команд на языке ассемблера	6
-----	---	---

1 Цель работы

Приобрести навыки работы с арифметическими инструкциями языка ассемблера NASM.

2 Теоретическое введение

Большая часть инструкций на языке ассемблера требует обработки операндов. Адрес операнда - это место, где хранятся данные, которые подлежат обработке.

Существует три основных способа адресации: 1. Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`. 2. Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`. 3. Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти.

Таблица 2.1: Описание некоторых команд на языке ассемблера

Команда	Арифметическая операция
<code>add</code>	Целочисленное сложение
<code>sub</code>	Целочисленное вычитание
<code>neg</code>	Изменение знака операнда
<code>mul</code>	Умножение
<code>div</code>	Деление

3 Выполнение лабораторной работы

При помощи команды `mkdir` создаю папку для работы (рис. 3.1).

```
aobudnik@dk8n64 ~ $ mkdir ~/work/arch-pc/lab06
```

Рис. 3.1: Создание директории

Перехожу в созданный каталог `~/work/arch-pc/lab06` и создаю файл `lab6-1.asm` (рис. 3.2).

```
aobudnik@dk8n64 ~ $ cd ~/work/arch-pc/lab06
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ touch lab6-1.asm
```

Рис. 3.2: Создание файла

Ввожу код программы из листинга 6.1 для вывода значения регистра `eax` (рис. 3.3).

```
lab6-1.asm [-M--] 12 L:[ 1+16 17/ 17] *(210 / 210b) <EOF> [*][X]
#include "in_out.asm"

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

    mov eax, '0'
    mov ebx, '4'
    add eax, ebx
    mov [buf1], eax
    mov eax, buf1
    call sprintf
    call quit
```

Рис. 3.3: Код программы lab6-1

Создаю объектный файл и с помощью компоновщика LD обрабатываю его. Получаю исполняемый файл. Запускаю программу (рис. 3.4).

```
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-1
j
```

Рис. 3.4: Запуск программы

Изменяю код в соответствии с заданием (рис. 3.5).

```
lab6-1.asm  [-M--] 12 L: [ 1+16 17/ 17] *(206 / 206b) <EOF>  [*][X]
#include "incout.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. 3.5: Изменение кода

Создаю объектный файл и с помощью компоновщика LD обрабатываю его. Получаю исполняемый файл. Запускаю программу. Полученный результат - символ перехода на новую строку (рис. 3.6).

```
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-1
```

Рис. 3.6: Запуск программы

Создаю файл lab6-2.asm (рис. 3.7).


```
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ touch lab6-2.asm
```

Рис. 3.7: Создание файла

Ввожу код программы из листинга 6.2 для вывода значения регистра `eax` (рис. 3.8).

```
lab6-2.asm [-M--] 11 L:[ 1+11 12/ 12] [*][X]
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF

call quit
```

Рис. 3.8: Код программы lab6-2

Создаю объектный файл и с помощью компоновщика LD обрабатываю его. Получаю исполняемый файл. Запускаю программу (рис. 3.9).

```
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 3.9: Запуск программы

Изменяю код программы, меняя символы на числа (рис. 3.10).

```
lab6-2.asm [-M--] 2 L:[ 1+12 13/ 13] *(137 / 137b) <EOF> [*][X]
#include "lab6-2.asm"

SECTION .text
GLOBAL _start
_start:

mov eax,6
mov ebx,4
add eax,ebx
call iprintLF

call quit
```

Рис. 3.10: Изменение кода

Произвожу трансляцию и линковку. Запускаю получившуюся программу (рис. 3.13).

```
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 3.11: Запуск программы

Изменяю iprintLF на iprint в тексте программы. (рис. 3.12).

```
lab6-2.asm [-M--] 13 L:[ 1+ 9 10/ 13] *(117 / 135b) 0010 0x00A [*][X]
#include "lab6-2.asm"

SECTION .text
GLOBAL _start
_start:

mov eax,6
mov ebx,4
add eax,ebx
call iprint

call quit
```

Рис. 3.12: Изменение кода

Произвожу трансляцию и линковку. Запускаю получившуюся программу (рис. ??). Команда iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

```

aobudnik@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-2
10aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-2
10aobudnik@dk8n64 ~/work/arch-pc/lab06 $

```

Рис. 3.13: Запуск программы

В каталоге создаю файл lab6-3.asm (рис. 3.14).

```

aobudnik@dk8n64 ~/work/arch-pc/lab06 $ touch lab6-3.asm

```

Рис. 3.14: Создание файла

Ввожу код программы из листинга 6.3 для вычисления выражения $\text{div}(5) = (5 \times 2 + 3)/3$ (рис. 3.15).

```

lab6-3.asm  [-M--] 11 L: [ 1+31 32/ 32] *(404 / 404b) <EOF>  [*][X]
#include "input.asm"

SECTION .data
div: DB "Division: ",0
rem: DB "Remainder of division: ",0

SECTION .text
GLOBAL _start
_start:

mov eax,5
mov ebx,2
mul ebx
add eax,3
xor edx,edx
mov ebx,3
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. 3.15: Код программы lab6-3

Создаю объектный файл и обрабатываю его при помощи компоновщика. Получаю исполняемый файл и запускаю программу (рис. 3.16).

```

aobudnik@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1

```

Рис. 3.16: Запуск программы

Изменяю код программы так, чтобы она вычисляла значение выражения $\boxed{4}(\boxed{6})$
 $= (4 \cdot 6 + 2)/5$. (рис. 3.17).

```

lab6-3.asm  [-M--] 11 L: [ 1+16 17/ 32] *(240 / 404b) 0010 0x00A  [*][X] ^
#include "lab6-3.asm"

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

SECTION .text
GLOBAL _start
_start:

mov eax,6
mov ebx,4
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. 3.17: Изменение кода

Произвожу трансляцию и линковку. Запускаю получившуюся программу (рис. 3.18).

```

aobudnik@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1

```

Рис. 3.18: Запуск программы

Создаю файл variant.asm при помощи утилиты touch и ввожу в нее код про-

граммы из листинга 6.4 для вычисления варианта задания по номеру студенческого билета (рис. 3.19).

```
variant.asm [-M--] 11 L:[ 1+33 34/ 34] *(444 / 444b) <E[*][X]
#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 printf

    mov ecx, x
    mov edx, 80
    call read

    mov eax, x
    call atoi

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

    mov eax, rem
    call printf
    mov eax, edx
    call printf

    call quit
```

Рис. 3.19: Код программы variant

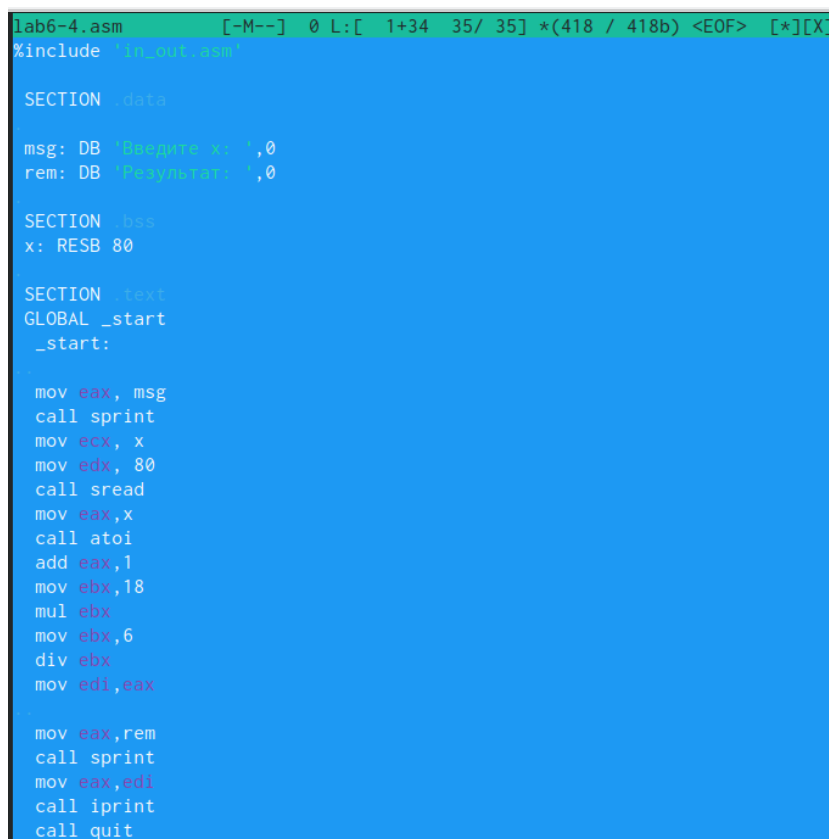
Произвожу трансляцию и линковку. Запускаю получившуюся программу (рис. 3.20). Мой вариант под номером 17.

```
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
aobudnik@dk8n64 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132236076
Ваш вариант: 17
```

Рис. 3.20: Запуск программы

4 Выполнение заданий для самостоятельной работы

При помощи команды touch создаю файл lab6-4.asm. Ввожу код программы для вычисления выражения $18(x + 1)/6$ относительно переменной x (рис. 4.1).



```
lab6-4.asm [-M--] 0 L: [ 1+34 35/ 35] *(418 / 418b) <EOF> [*][X]
%include "in_out.asm"

SECTION .data
msg: DB "Введите x: ",0
rem: DB "Результат: ",0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
add eax, 1
mov ebx, 18
mul ebx
mov ebx, 6
div ebx
mov edi, eax

mov eax, rem
call sprint
mov eax, edi
call iprint
call quit
```

Рис. 4.1: Код программы lab6-4

Создаю объектный файл и обрабатываю его при помощи компоновщика. По-

лучаю исполняемый файл (рис. 4.2).

```
lab6: ld: error: symbol `x' not defined
aobudnik@dk8n70 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
aobudnik@dk8n70 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
```

Рис. 4.2: Линковка

Запускаю программу и проверяю результат для значений x1 и x2 (рис. 4.3).

```
aobudnik@dk8n70 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
aobudnik@dk8n70 ~/work/arch-pc/lab06 $ ./lab6-4
Введите x: 3
Результат: 12aobudnik@dk8n70 ~/work/arch-pc/lab06 $ ./lab6-4
Введите x: 1
aobudnik@dk8n70 ~/work/arch-pc/lab06 $
```

Рис. 4.3: Запуск программы

Все созданные в процессе работы файлы копирую в локальный репозиторий и загружаю на GitHub.

5 Выводы

По итогам выполнения лабораторной работы №6 я научилась работать с некоторыми арифметическими инструкциями языка ассемблера NASM.