

Отчёт по лабораторной работе №6

**Дисциплина: Компьютерные технологии и технологии
программирования**

ДЫМОВОЙ Д.Д.

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выводы	14
	Список литературы	15

Список иллюстраций

3.1	Создание каталога и файла asm	7
3.2	Листинг 6.1	7
3.3	Компановка и трансляция	7
3.4	Запуск исполняемого файла	8
3.5	Листинг 6.2	8
3.6	Компановка и трансляция	8
3.7	Запуск исполняемого файла	9
3.8	Запуск исполняемого файла	9
3.9	Использование команды touch	9
3.10	Листинг 6.3	10
3.11	Компановка и трансляция	10
3.12	Запуск исполняемого файла	10
3.13	Создание файла	11
3.14	Листинг 6.4	11
3.15	Компановка и трансляция	11
3.16	Программа	13

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Теоретическое введение

Адресация в NASM

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.

Арифметические операции в NASM:

`add` , Целочисленное сложение `sub` , Целочисленное вычитание

3 Выполнение лабораторной работы

Я создаю каталог для программ лабораторной работы №6, перехожу в него и создаю файл lab6-1.asm (рис. 3.1).

```
dddihmo@dk8n76 ~ $ mkdir ~/work/arch-pc/lab06
dddihmo@dk8n76 ~ $ cd ~/work/arch-pc/lab06
dddihmo@dk8n76 ~/work/arch-pc/lab06 $ touch lab6-1.asm
```

Рис. 3.1: Создание каталога и файла asm

Ввожу текст программы из листинга 6.1 в файл lab6-1.asm (рис. 3.2).

```
lab6-1.asm [----] 4 L: [ 1+15 16/ 17] *(206 / 220b) 0010 0x00A
#include "lab_out.asm"

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, 10
mov ebx, 4
add eax, ebx
mov [buf1], eax
mov eax, buf1
call sprintf
call quit
```

Рис. 3.2: Листинг 6.1

Создаю исполняемый файл и запускаю его (рис. 3.3).

```
dddihmo@dk8n76 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
dddihmo@dk8n76 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
dddihmo@dk8n76 ~/work/arch-pc/lab06 $ ./lab6-1
j
```

Рис. 3.3: Компиляция и трансляция

Вношу изменения согласно заданию и ещё раз создаю исполняемый файл, запускаю его (рис. 3.4).

```
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ./lab6-1
```

Рис. 3.4: Запуск исполняемого файла

Согласно таблице ASCII код 10 соответствует символу STX, этот символ не отображается.

Создаю с помощью команды touch файл lab6-2.asm и ввожу текст программы из листинга 6.2 (рис. 3.5).

```
lab6-2.asm [-M--] 13 L:[ 1+11 12/ 12] *(160 / 160
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

    call quit
```

Рис. 3.5: Листинг 6.2

Создаю исполняемый файл и запускаю его (рис. 3.6).

```
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 3.6: Компиляция и трансляция

Вношу изменения согласно заданию и ещё раз создаю исполняемый файл, запускаю его (рис. 3.7).


```

dddihmova@dk8n76 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ./lab6-2
10

```

Рис. 3.7: Запуск исполняемого файла

Результат число 10.

Заменяю функцию `iprintLF` на `iprint`, снова создаю исполняемый файл и запускаю (рис. 3.8).

```

dddihmova@dk8n76 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ./lab6-2
10dddihmova@dk8n76 ~/work/arch-pc/lab06 $

```

Рис. 3.8: Запуск исполняемого файла

Функции `iprintLF` и `iprint` отличаются тем, что `iprint` выводит в той же строке, а `iprintLF` переходит на новую.

Создаю файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06` (рис. 3.9).

```

dddihmova@dk8n76 ~ $ touch ~/work/arch-pc/lab06/lab6-3.asm

```

Рис. 3.9: Использование команды `touch`

Ввожу текст программы из листинга 6.3 в созданный файл `lab6-3.asm` (рис. 3.10).

```

lab6-3.asm      [-M--]  4 L: [ 1+30  31/ 32] *(1396/1469b) 0010 0x00A
;~~~~~
; Программа вычисления выражения
;~~~~~
%include "lab_out.asm" ; подключение внешнего файла
SECTION .data
div: DB "Результат: ",0
rem: DB "Остаток от деления: ",0

SECTION .text
GLOBAL _start
----- _start:
; ---- Вычисление выражения
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'
; ---- 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. 3.10: Листинг 6.3

Создаю исполняемый файл и запускаю его (рис. 3.11).

```

dddihmova@dk8n76 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1

```

Рис. 3.11: Компиляция и трансляция

Вношу изменения согласно заданию и ещё раз создаю исполняемый файл, запускаю его (рис. ??).

```

dddihmova@dk8n76 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1

```

Рис. 3.12: Запуск исполняемого файла

С помощью команды touch создаю файл variant.asm (рис. 3.13).

```
dddihmovaldk8n76 ~/work/arch-pc/lab06 $ touch variant.asm
```

Рис. 3.13: Создание файла

Ввожу текст программы из листинга 6.4 в созданный файл variant.asm (рис. 3.14).

```
variant.asm [----] 11 L: [ 1+32 33/ 36] *(644 / 669b) 0010 0x00A
; 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 sprintLF
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 sprint
mov eax, edx
call iprintLF
call quit
```

Рис. 3.14: Листинг 6.4

Создаю исполняемый файл и запускаю его (рис. 3.15).

```
dddihmovaldk8n76 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
dddihmovaldk8n76 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
dddihmovaldk8n76 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132236092
Ваш вариант: 13
```

Рис. 3.15: Компиляция и трансляция

Ответы на вопросы:

- 1) Сообщение 'Ваш вариант:' сначала задаётся переменной (rem: DB 'Ваш вариант:', 0), а затем выводится с помощью команд:

```
mov eax, msg call sprintLF
```

- 2) В переменную x заносится вводимый текст. Команда mov edx, 80 записывает длину вводимого сообщения, call sread отвечает за вызов подпрограммы ввода сообщения.
- 3) эта функция преобразует ASCII-код символа в целое число и записывает результат в регистр eax.
- 4) Переменной msg задаётся номер студенческого. А переменной rem полученный вариант.

```
msg: DB 'Введите No студенческого билета:', 0 rem: DB 'Ваш вариант:', 0
```

Далее call sprintLF выводит сообщение 'Введите No студенческого билета:' на экран, производится ввод с клавиатуры. Вызывается подпрограмма (call sread) отвечающая за ввод сообщения.

Далее обнуляется edx для корректной работы div:

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

В edx записывается результат.

Далее вызывается с помощью подпрограммы переменная rem и выводится посчитанный ответ.

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

Завершение работы программы.

```
call quit
```

- 5) В регистр edx
- 6) Это команда инкремента, она увеличивает значение регистра на 1.

7)Результат вычислений: mov eax,rem call sprint mov eax,edx call iprintLF #

Задание для самостоятельной работы

Я написала программу для вычисления выражения из 13 варианта (рис. 3.16).

```
/afs/.dk.sci.pfu.edu.ru/home/d/d/ddihmova/work/arch-pc/lab06/zadanie.asm
#include 'in_out.asm'

SECTION .data
fyn: DB 'Вычислите: y=(8x+6)*10',0
per: DB 'Введите x: ',0
otw: DB 'Результат:',0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, fyn
call sprintLF

mov eax, per
call sprintLF

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

mov ebx, 8
mul ebx

add eax, 6

mov ebx, 10
mul ebx

mov edi, eax

mov eax, otw
call sprintLF
mov eax, edi
call iprintLF

call quit
```

Рис. 3.16: Программа

Делаю компановку и трансляцию, провожу проверку работы программы для

```
ddihmova@dk8n62 ~/work/arch-pc/lab06 $ nasm -f elf zadanie.asm
ddihmova@dk8n62 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o zadanie zadanie.o
ddihmova@dk8n62 ~/work/arch-pc/lab06 $ ./zadanie
Вычислите: y=(8x+6)*10
Введите x:
1
Результат:
140
ddihmova@dk8n62 ~/work/arch-pc/lab06 $ ./zadanie
Вычислите: y=(8x+6)*10
Введите x:
4
Результат:
380
```

x=1 и x=4(рис. ??).

4 Выводы

Я освоила арифметические инструкции языка ассемблера NASM.

Список литературы