Отчёт по лабораторной работе №6

Дисциплина: Комьютерные технологии и технологии программирования

Дымовой Д.Д.

Содержание

Список иллюстраций

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 ко- манде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символи- ческое 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).

```
dddihmova@dk8n76 ~ $ mkdir ~/work/arch-pc/lab06
dddihmova@dk8n76 ~ $ cd ~/work/arch-pc/lab06
dddihmova@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 '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 sprintLF
    call quit
```

Рис. 3.2: Листинг 6.1

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

```
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
j
```

Рис. 3.3: Компановка и трансляция

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

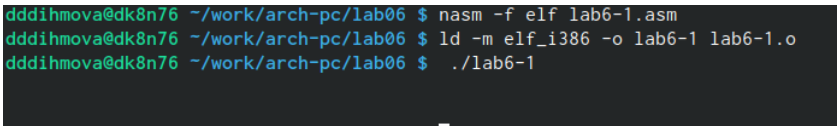

Рис. 3.4: Запуск исполняемого файла

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).

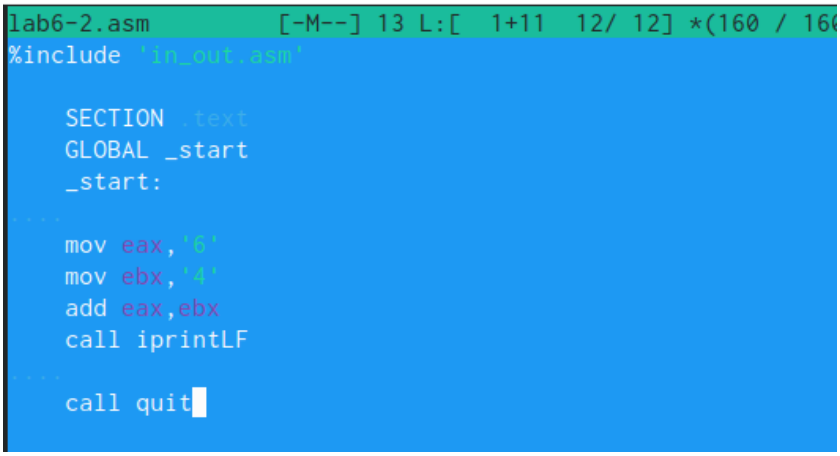

Рис. 3.5: Листинг 6.2

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

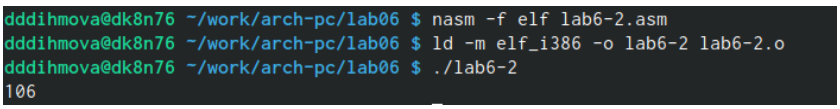

Рис. 3.6: Компановка и трансляция

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

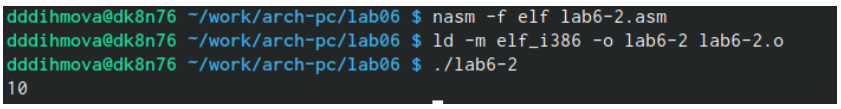

Рис. 3.7: Запуск исполняемого файла

Результат число 10.

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

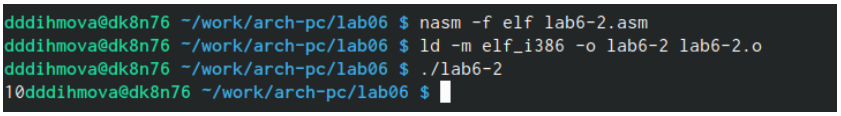

Рис. 3.8: Запуск исполняемого файла

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).

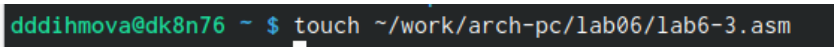

Рис. 3.9: Использование команды touch

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

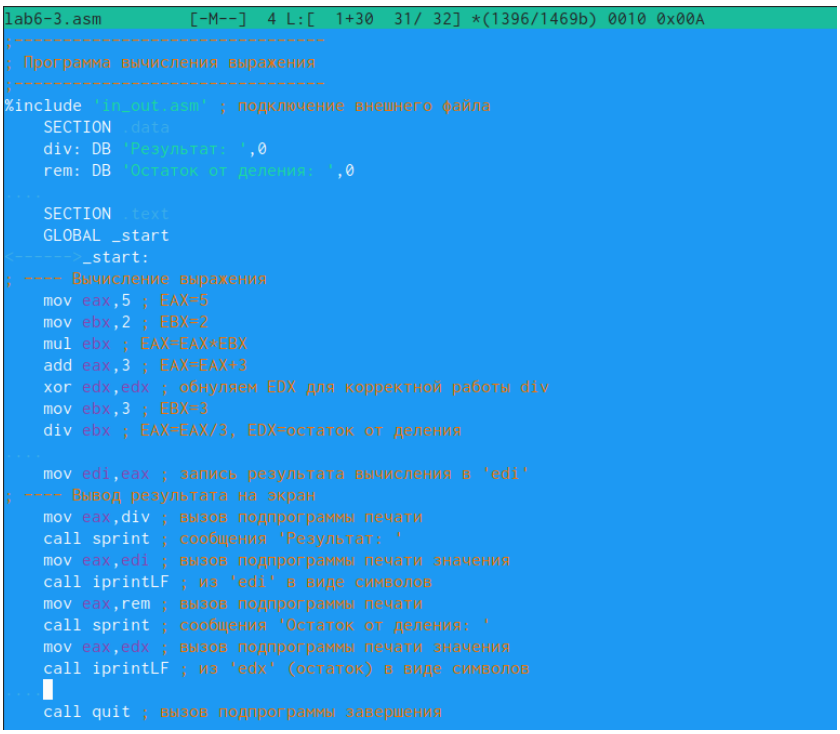

Рис. 3.10: Листинг 6.3

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

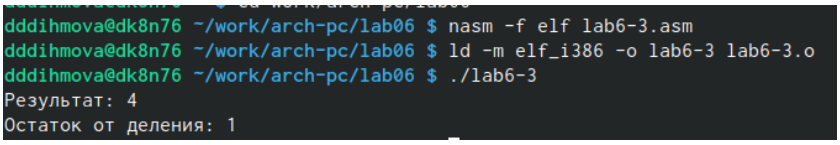

Рис. 3.11: Компановка и трансляция

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

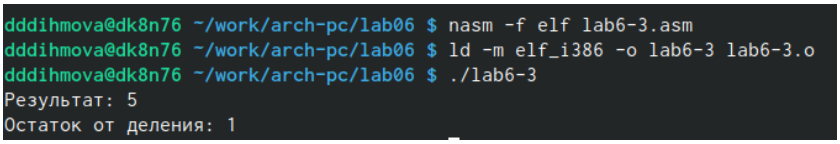

Рис. 3.12: Запуск исполняемого файла

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

```
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ touch variant.asm
```

Рис. 3.13: Создание файла

Ввожу текст программый 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).

```
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
dddihmova@dk8n76 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132236092
Ваш вариант: 13
```

Рис. 3.15: Компановка 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

Завершенние 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/dddihmova/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 x=1 и x=4(рис. **??**).

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

4 Выводы

Я освоила арифметические инструкции языка ассемблера NASM.

Список литературы