

Отчет по выполнению лабораторной работы №6

Дисциплина: архитектура компьютеров

Намруев Максим Саналович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Символьные и численные данные в NASM	8
4.2	Выполнение арифметических операций в NASM	12
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Создание каталога	8
4.2	Ввод программы из листинга 6.1	9
4.3	Запуск файла	9
4.4	Исправление программы	9
4.5	Запуск исправленной программы	10
4.6	Создание нового файла	10
4.7	Запуск новой программы	10
4.8	Изменение программы	11
4.9	Запуск исправленной программы	11
4.10	Изменение файла	11
4.11	Запуск программы	12
4.12	Создание файла	12
4.13	Ввод программы из листинга 6.3	13
4.14	Запуск программы	13
4.15	Редактирование текста программы	14
4.16	Запуск программы	14
4.17	Создание файла variant.asm	15
4.18	Создание файла variant.asm	15
4.19	Запуск файла	16
4.20	Создание файла	16
4.21	Ввод программы	17
4.22	Проверка для значения x1	17
4.23	Проверка для значения x2	17

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM

3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти.

Регистровая адресация – операнды хранятся в регистрах и в команде используются имена регистров.

Непосредственная адресация – значение операнда задается непосредственно в команде.

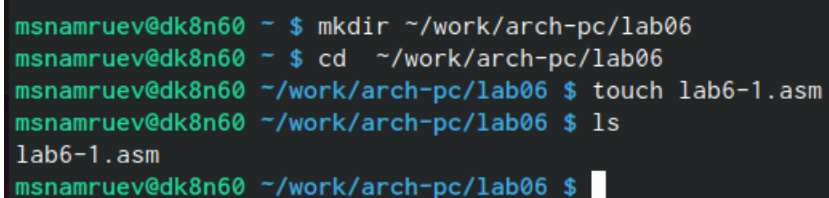
Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое имя или адрес.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

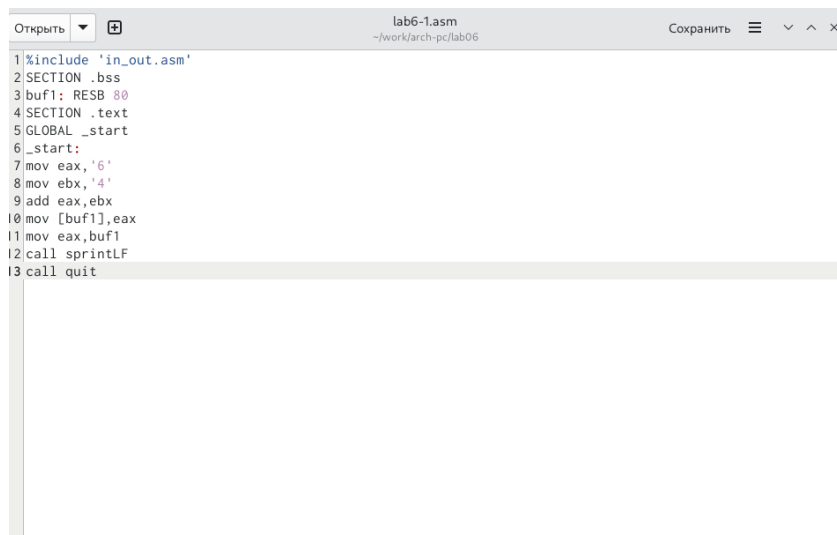
Создаю каталог для программ лабораторной работы №6, перехожу в него и создаю файл lab6-1.asm (рис. [4.1]).



```
msnamruev@dk8n60 ~ $ mkdir ~/work/arch-pc/lab06
msnamruev@dk8n60 ~ $ cd ~/work/arch-pc/lab06
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ touch lab6-1.asm
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ ls
lab6-1.asm
msnamruev@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 4.1: Создание каталога

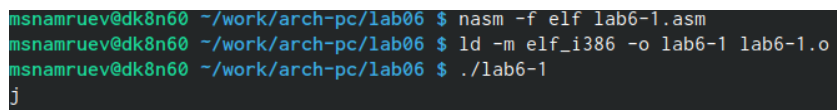
Ввожу в файл lab6-1.asm текст программы из листинга 6.1.(рис. [4.2]).



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

Рис. 4.2: Ввод программы из листинга 6.1

Создаю исполняемый файл и запускаю его.(рис. [4.3]).



```
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-1
j
```

Рис. 4.3: Запуск файла

Изменяю текст программы и вместо символов записываю в регистры числа.(рис. [4.4]).



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

Рис. 4.4: Исправление программы

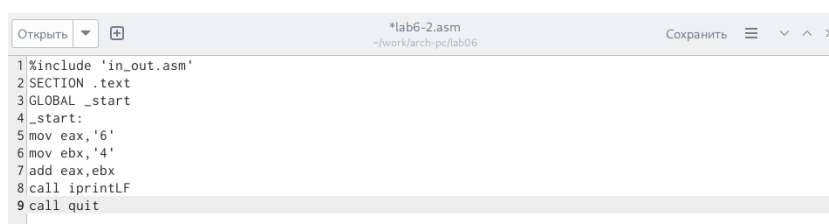
Далее создаю исполняемый файл и запускаю его.(рис. [4.5]).

```
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-1
```

Рис. 4.5: Запуск исправленной программы

Пользуясь таблицей ASCII можно определить, что код 10 соответствует символу переносу строки. Этот символ не отображается на экране.

Создаю новый файл lab6-2 в том же каталоге и ввожу в него текст программы из листинга 6.2.(рис. [4.6]).



```
Открыть  *lab6-2.asm  Сохранить
~/work/arch-pc/lab06
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, '6'
6 mov ebx, '4'
7 add eax, ebx
8 call iprintLF
9 call quit
```

Рис. 4.6: Создание нового файла

Создаю исполняемый файл и запускаю его.(рис. [4.7]).

```
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ ls -m elf_i386 -o lab6-2 lab6-2.o
ls: невозможно получить доступ к 'elf_i386': Нет такого файла или каталога
ls: невозможно получить доступ к 'lab6-2': Нет такого файла или каталога
-rw-r--r-- 1 msnamruev 1040 ноя 14 12:31 lab6-2.o
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 4.7: Запуск новой программы

Аналогично предыдущему примеру изменяю символы на числа.(рис. [4.8]).

```
lab6-2.asm [----] 9 L: [ 1+ 5 6/ 10] *(77 / 114b) 0010 0x00A
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 4.8: Изменение программы

Создаю исполняемый файл и запускаю его.(рис. [4.9]).

```
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
msnamruev@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 4.9: Запуск исправленной программы

При исполнении программы было получено число 10.

Заменяю функцию iprintLF на iprint. Создаю исполняемый файл и запускаю его.(рис. [4.10]).(рис. [4.11]).

```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprint
9 call quit
```

Рис. 4.10: Изменение файла

```
msnamruev@dk8n70 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
msnamruev@dk8n70 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
msnamruev@dk8n70 ~/work/arch-pc/lab06 $ ./lab6-2
10msnamruev@dk8n70 ~/work/arch-pc/lab06 $
```

Рис. 4.11: Запуск программы

Вывод функции `iprint` отличается от `iprintLF` тем, что выведенное сообщение не переносится на следующую строку.

4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06`. (рис. [4.12]).

```
msnamruev@dk5n60 ~/work/arch-pc/lab06 $ touch lab6-3.asm
msnamruev@dk5n60 ~/work/arch-pc/lab06 $
```

Рис. 4.12: Создание файла

После внимательного прочтения текста программы из листинга 6.3 ввожу его в `lab6-3.asm`. (рис. [4.13]).

```

lab6-3.asm [----] 33 L: [ 1+ 0 1/ 29] *(33 /1365b) 0010 0x00A
; Программа вычисления выражения
%include 'in_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

```

Рис. 4.13: Ввод программы из листинга 6.3

Создаю исполняемый файл и запускаю его.(рис. [4.14]).

```

msnamruev@dk8n70 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
msnamruev@dk8n70 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
msnamruev@dk8n70 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
msnamruev@dk8n70 ~/work/arch-pc/lab06 $

```

Рис. 4.14: Запуск программы

Изменяю тест программы для вычисления выражения $f(x)=(4*6+2)/5$.(рис. [4.15]).

```

1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4 %include 'in_out.asm' ; подключение внешнего файла
5 SECTION .data
6 div: DB 'Результат: ',0
7 rem: DB 'Остаток от деления: ',0
8 SECTION .text
9 GLOBAL _start
0 _start:
1 ; ---- Вычисление выражения
2 mov eax,4 ; EAX=4
3 mov ebx,6 ; EBX=6
4 mul ebx ; EAX=EAX*EBX
5 add eax,2 ; EAX=EAX+2
6 xor edx,edx ; обнуляем EDX для корректной работы div
7 mov ebx,5 ; EBX=5
8 div ebx ; EAX=EAX/5, EDX=остаток от деления
9 mov edi,eax ; запись результата вычисления в 'edi'
0 ; ---- Вывод результата на экран
1 mov eax,div ; вызов подпрограммы печати
2 call sprint ; сообщения 'Результат: '
3 mov eax,edi ; вызов подпрограммы печати значения
4 call iprintLF ; из 'edi' в виде символов
5 mov eax,rem ; вызов подпрограммы печати
6 call sprint ; сообщения 'Остаток от деления: '
7 mov eax,edx ; вызов подпрограммы печати значения
8 call iprintLF ; из 'edx' (остаток) в виде символов
9 call quit ; вызов подпрограммы завершения

```

Рис. 4.15: Редактирование текста программы

Создаю исполняемый файл и запускаю его.(рис. [4.16]).

```

msnamruev@dk8n70 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
msnamruev@dk8n70 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
msnamruev@dk8n70 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
msnamruev@dk8n70 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
msnamruev@dk8n70 ~/work/arch-pc/lab06 $

```

Рис. 4.16: Запуск программы

Создаю файл variant.asm в каталоге ~/work/arch-pc/lab06.(рис. [4.17]).

```
msnamruev@dk5n60 ~/work/arch-pc/lab06 $ touch variant.asm
msnamruev@dk5n60 ~/work/arch-pc/lab06 $
```

Рис. 4.17: Создание файла variant.asm

Внимательно изучаю текст программы из листинга 6.4 и ввожу в файл variant.asm.(рис. [4.19]).

```

1 ;-----
2 ; Программа вычисления варианта
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg: DB 'Введите No студенческого билета: ',0
7 rem: DB 'Ваш вариант: ',0
8 SECTION .bss
9 x: RESB 80
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprintLF
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax,x ; вызов подпрограммы преобразования
19 call atoi ; ASCII кода в число, 'eax=x
20 xor edx,edx
21 mov ebx,20
22 div ebx
23 inc edx
24 mov eax,rem
25 call sprint
26 mov eax,edx
27 call iprintLF
28 call quit

```

Рис. 4.18: Создание файла variant.asm

Создаю исполняемый файл и запускаю его.(рис. [4.19]).

```

msnamruev@dk8n70 ~/work/arch-pc/lab06 $ gedit variant.asm
msnamruev@dk8n70 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
msnamruev@dk8n70 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
msnamruev@dk8n70 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132236035
Ваш вариант: 16
msnamruev@dk8n70 ~/work/arch-pc/lab06 $

```

Рис. 4.19: Запуск файла

Ответы на вопросы

1. За вывод сообщение “Ваш вариант” отвечают строки: `mov eax,rem call sprint`
2. Эти строки используются чтобы считать х.
3. `Call atoi` преобразовывает код ASCII в целое число
4. За вычисление варианта отчедают строки: `xor edx,edx mov ebx,20 div ebx inc edx`
5. Остаток от деление записывается в регистр `edx`.
6. Инструкция `inc edx` используется для того, чтобы увеличить значение регистра `edx` на 1.
7. Для вывода на экран результата выислений используются строки: `mov eax,edx call iprintLF`

#Задание для самостоятельной работы

Создаю файл `lab6-4.asm` в каталоге `~/work/arch-pc/lab06`. Проверяю создание файла.(рис. [4.20]).

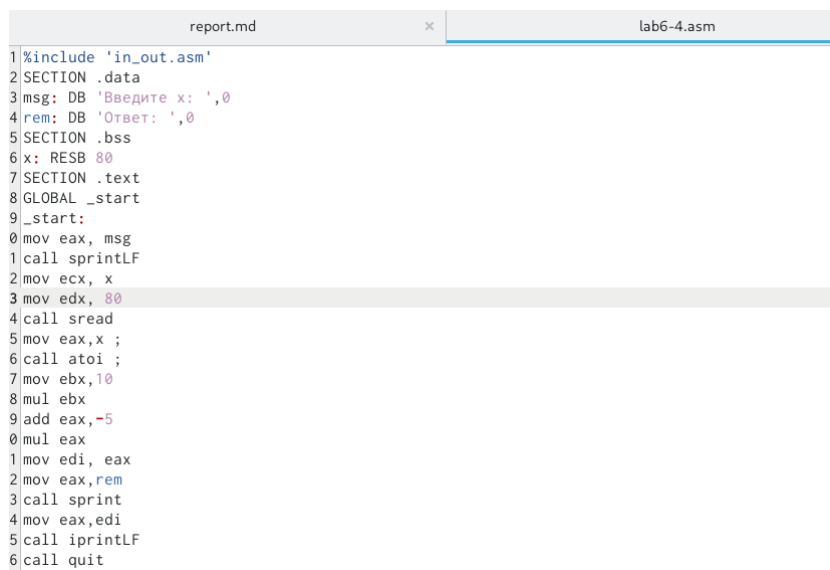
```

msnamruev@dk5n60 ~/work/arch-pc/lab06 $ touch lab6-4.asm
msnamruev@dk5n60 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm  lab6-4.asm  variant.asm
lab6-1      lab6-1.o    lab6-2.asm  lab6-3    lab6-3.o    variant     variant.o
msnamruev@dk5n60 ~/work/arch-pc/lab06 $

```

Рис. 4.20: Создание файла

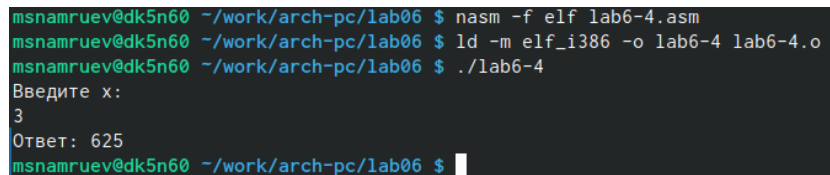
Открываю созданную файл и начинаю печатать в него текст программы для вычисления $(10 \cdot x - 5)^2$ (вариант 16)(рис. [4.21]).



```
report.md x lab6-4.asm
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 rem: DB 'Ответ: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x ;
16 call atoi ;
17 mov ebx, 10
18 mul ebx
19 add eax, -5
20 mul eax
21 mov edi, eax
22 mov eax, rem
23 call sprintf
24 mov eax, edi
25 call iprintLF
26 call quit
```

Рис. 4.21: Ввод программы

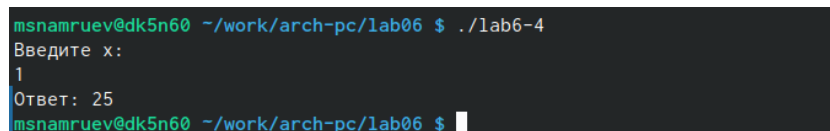
Далее сохраню файл, создаю исполняемый файл и запускаю его. (рис. [4.22]).



```
msnamruev@dk5n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
msnamruev@dk5n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
msnamruev@dk5n60 ~/work/arch-pc/lab06 $ ./lab6-4
Введите x:
3
Ответ: 625
msnamruev@dk5n60 ~/work/arch-pc/lab06 $
```

Рис. 4.22: Проверка для значения x1

Запускаю программу ещё раз для проверки её работы при x2.(рис. [4.23]).



```
msnamruev@dk5n60 ~/work/arch-pc/lab06 $ ./lab6-4
Введите x:
1
Ответ: 25
msnamruev@dk5n60 ~/work/arch-pc/lab06 $
```

Рис. 4.23: Проверка для значения x2

Всё верно работает. Сама программа:

```
%include 'in_out.asm' SECTION .data msg: DB 'Введите x:',0 rem: DB 'Ответ:',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 ; call atoi ; mov ebx,10 mul ebx
add eax,-5 mul eax mov edi, eax mov eax,rem call sprint mov eax,edi call iprintLF call
quit
```

5 Выводы

После выполнения данной работы я освоил арифметические инструкции языка ассемблера NASM

Список литературы