

Отчет пол лабораторной работе №6

Простейший вариант

Михальский Кирилл Алексеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выполнение задания для самостоятельной работы.	13
6	Выводы	14
	Список литературы	15

Список иллюстраций

4.1	Создание	8
4.2	Изменение	8
4.3	Компиляция	9
4.4	Запуск	9
4.5	Создание	10
4.6	Изменения	10
4.7	Изменения	10
4.8	Создание и ввод	10
4.9	Запуск	11
4.10	Изменение и запуск	11
4.11	Работа с файлом	11
5.1	Работа с файлом	13
5.2	Работа с файлом	13

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

2 Задание

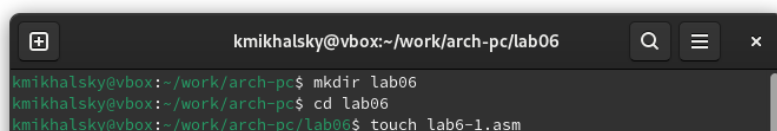
1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM

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. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. 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.

4 Выполнение лабораторной работы

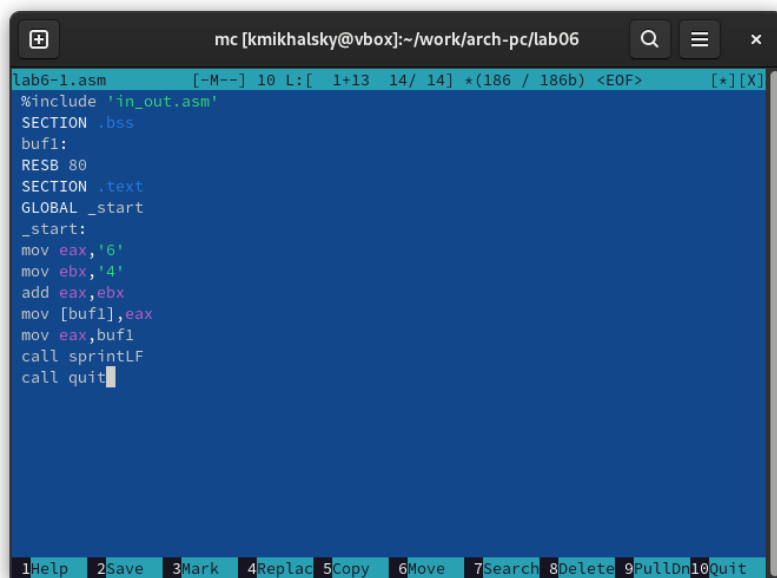
1. Создал каталог и файл lab6-1.asm.



```
kmikhalsky@vbox:~/work/arch-pc/lab06
kmikhalsky@vbox:~/work/arch-pc$ mkdir lab06
kmikhalsky@vbox:~/work/arch-pc$ cd lab06
kmikhalsky@vbox:~/work/arch-pc/lab06$ touch lab6-1.asm
```

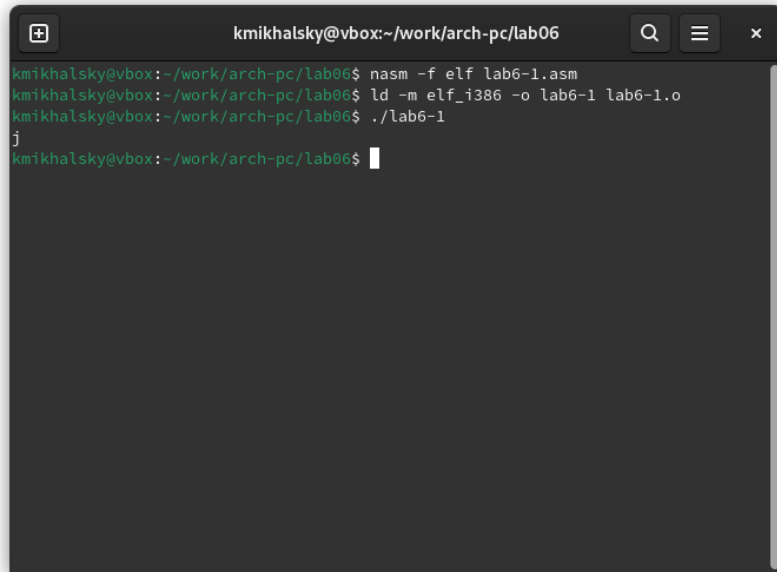
Рис. 4.1: Создание

2. Вставил текст программы и запустил.



```
lab6-1.asm [-M--] 10 L: [ 1+13 14/ 14] *(186 / 186b) <EOF> [*] [X]
#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 sprintf
call quit
```

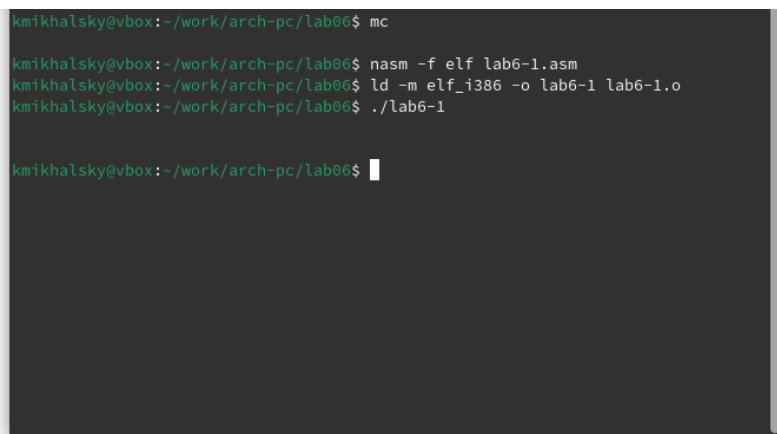
Рис. 4.2: Изменение



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

Рис. 4.3: Компиляция

3. Внес необходимые изменения и запустил. Код соответствует пустому символу перевода строки, который и отображался на экране.



```
kmikhalsky@vbox:~/work/arch-pc/lab06$ mc
kmikhalsky@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
kmikhalsky@vbox:~/work/arch-pc/lab06$ ./lab6-1
j
kmikhalsky@vbox:~/work/arch-pc/lab06$
```

Рис. 4.4: Запуск

4. Создал файл lab6-2.asm, вставил текст программы и запустил исполняемый файл.

```

kmikhalsky@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
kmikhalsky@vbox:~/work/arch-pc/lab06$ ./lab6-2
106
kmikhalsky@vbox:~/work/arch-pc/lab06$

```

Рис. 4.5: Создание

- Внес изменения в файл lab6-2.asm. $6 + 4 = 10$. Заменял функцию `iprintLF` на `iprint` и запустил исполняемый файл. Не отобразился перевод строки.

```

kmikhalsky@vbox:~/work/arch-pc/lab06$ gedit lab6-2.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
kmikhalsky@vbox:~/work/arch-pc/lab06$ ./lab6-2
10kmikhalsky@vbox:~/work/arch-pc/lab06$

```

Рис. 4.6: Изменения

```

kmikhalsky@vbox:~/work/arch-pc/lab06$ gedit lab6-2.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
kmikhalsky@vbox:~/work/arch-pc/lab06$ ./lab6-2.asm
bash: ./lab6-2.asm: Permission denied
kmikhalsky@vbox:~/work/arch-pc/lab06$ ./lab6-2
10

```

Рис. 4.7: Изменения

- Создал файл lab6-3.asm. Ввел текст программы. Исправил текст программы.

```

kmikhalsky@vbox:~/work/arch-pc/lab06 — bash
kmikhalsky@vbox:~/work/arch-pc/lab06$ touch lab6-3.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ gedit lab6-3.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
lab6-3.asm:1: error: '%include' expects a file name
lab6-3.asm:2: error: label or instruction expected at start of line
kmikhalsky@vbox:~/work/arch-pc/lab06$ gedit lab6-3.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm

```

Рис. 4.8: Созданию и вводу

```
kmikhalsky@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
kmikhalsky@vbox:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
kmikhalsky@vbox:~/work/arch-pc/lab06$
```

Рис. 4.9: Запуск

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

Рис. 4.10: Изменение и запуск

7. Создал файл variant.asm, ввел текст и запустил.

```
kmikhalsky@vbox:~/work/arch-pc/lab06$ touch variant.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ gedit variant.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
kmikhalsky@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.0
ld: cannot find variant.0: No such file or directory
kmikhalsky@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
kmikhalsky@vbox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246748
Ваш вариант: 9
kmikhalsky@vbox:~/work/arch-pc/lab06$
```

Рис. 4.11: Работа с файлом

1. `mov eax,rem call sprint`
2. Инструкция `mov ecx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx` `mov edx, 80` - запись в регистр `edx` длины вводимой строки

call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры

3. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
4. xor edx,edx ; обнуление edx для корректной работы div mov ebx,20 ;
ebx = 20 div ebx ; eax = eax/20, edx - остаток от деления inc edx ;
edx = edx + 1
5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
6. Инструкция inc edx увеличивает значение регистра edx на 1
7. mov eax,edx call iprintLF

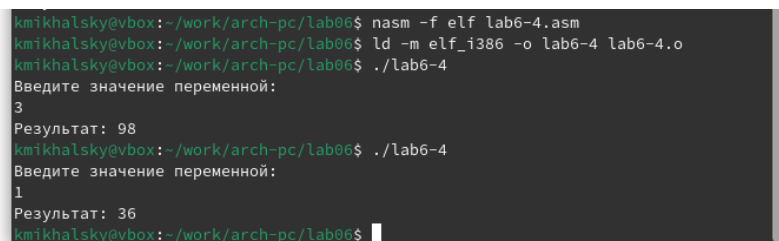
5 Выполнение задания для самостоятельной работы.

1. Копировал файл lab6-4, исправил текст программы для работы по условию, проверил правильность используя переменные из задачи. Вариант 9



```
1 include "io.h"
2 section .data
3 msg db "Введите значение переменной: ",0
4 fmt db "Результат: ",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 read
15 mov eax, x
16 call atoi
17 ; число программы преобразования
18 ; ASCII код в число, "eax"
19
20 mov ebx, 31
21 mul ebx
22 add ebx, -5
23 mov ebx, 10
24 add eax, ebx
25
26 mov edi, eax
27 mov eax, 0
28 call printf
29 mov eax, edi
30 call printf
31 call quit
```

Рис. 5.1: Работа с файлом



```
kmikhalsky@vbox: ~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
kmikhalsky@vbox: ~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
kmikhalsky@vbox: ~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной:
3
Результат: 98
kmikhalsky@vbox: ~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной:
1
Результат: 36
kmikhalsky@vbox: ~/work/arch-pc/lab06$
```

Рис. 5.2: Работа с файлом

6 Выводы

Я освоил арифметические конструкции языка ассемблера.

Список литературы