

Отчёт по лабораторной работе

Простейший вариант

Тимур Ринатович Каримов

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Создание каталога и файла	8
4.2	Написание программы	9
4.3	Выполнение программы	9
4.4	Изменение программы	10
4.5	Выполнение программы	10
4.6	Создание файла и написание программы	11
4.7	Выполнение программы	11
4.8	Изменение программы	12
4.9	Выполнение программы	12
4.10	Изменение программы	13
4.11	Выполнение программы	13
4.12	Создание файла и написание программы	14
4.13	Выполнение программы	14
4.14	Изменение программы	14
4.15	Выполнение программы	15
4.16	Создание файла и написание программы	15
4.17	Выполнение программы	15
4.18	Создание файла и написание программы	16
4.19	Выполнение программы	17

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . . .	7
-----	---	---

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

2 Задание

- 1) Выполнение лабораторной работы
- 2) Ответы на вопросы
- 3) Выполнение самостоятельной работы

3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы. Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux	
Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно про Unix см. в [1–4].

4 Выполнение лабораторной работы

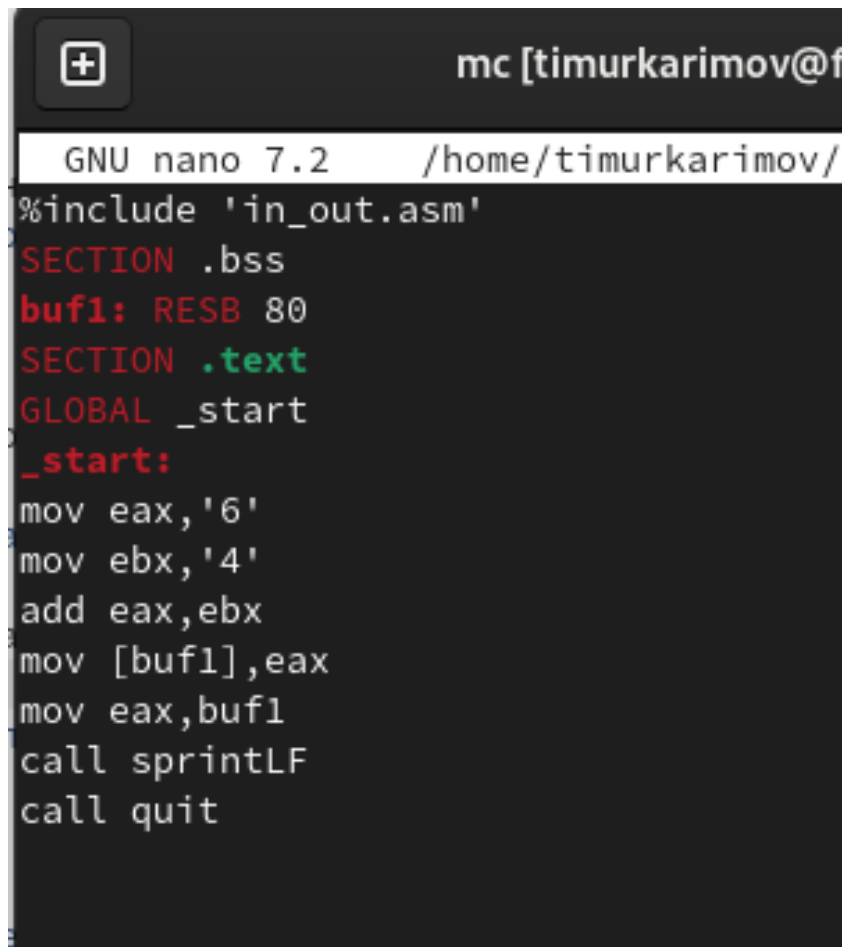
Создаем каталог для программ лабораторной №6. Переходим в него и создаем файл lab6-1.asm (рис. 4.1)

A terminal window with a dark background and green text. The text shows a series of commands being executed in a shell. The first command is 'mkdir ~/work/arch-pc/lab06'. The second command is 'cd ~/work/arch-pc/lab06'. The third command is 'touch lab6-1.asm'. The prompt changes from '~\$' to '~/work/arch-pc/lab06\$' after the 'cd' command.

```
существует
timurkarimov@fedora:~$ mkdir ~/work/arch-pc/lab06
timurkarimov@fedora:~$ cd ~/work/arch-pc/lab06
timurkarimov@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
```

Рис. 4.1: Создание каталога и файла

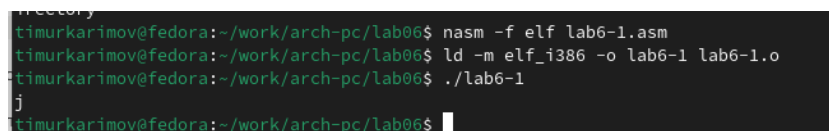
Ввожу в созданный файл текст программы из листинга (рис. 4.2)



```
mc [timurkarimov@f
GNU nano 7.2 /home/timurkarimov/
%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. 4.2: Написание программы

Создаем исполняемый файл и запускаем его(рис. 4.3)



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

Рис. 4.3: Выполнение программы

Изменяем текст программы, вместо символов записываем в `eax`, `ebx` числа (рис. 4.4)

```

GNU nano 7.2 /home
%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. 4.4: Изменение программы

Создаем исполняемый файл и запускаем его (рис. 4.5)

```

timurkarimov@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
timurkarimov@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
timurkarimov@fedora:~/work/arch-pc/lab06$ ./lab6-1

timurkarimov@fedora:~/work/arch-pc/lab06$

```

Рис. 4.5: Выполнение программы

На экране две пустые строки. Это связано с тем, что символ с кодом 10 - это символ перевода строки.

Создаем файл lab6-2.asm в каталоге для программ лабораторной №6. Ввожу в него текст программы из листинга 6.2 (рис. 4.6)

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

Рис. 4.6: Создание файла и написание программы

Создаем исполняемый файл и запускаем его (рис. 4.7)

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

Рис. 4.7: Выполнение программы

Аналогично предыдущей программе заменяем символы на числа (рис. 4.8)

```

%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

Создаем исполняемый файл и запускаем его (рис. 4.9)

```

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

```

Рис. 4.9: Выполнение программы

Теперь программа складывает не коды, соответствующие символам, а сами числа. Поэтому выводит число 10 - сумму чисел 4 и 6.

Заменяем функцию iprintLF на iprint (рис. 4.10)

```

%include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit

```

Рис. 4.10: Изменение программы

Создаем исполняемый файл и запускаем его(рис. 4.11)

```

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

```

Рис. 4.11: Выполнение программы

Вывод функции `iprintLF` от вывода функции `iprint` отличается тем, что в последнем случае после вывода не добавляется переход на новую строку.

2. Выполнение арифметических операций в Nasm

3. Создаем файл `lab6-3.asm`. Вводим в него текст программы для вычисления значения указанного выражения (рис. 4.12)

```

GNU nano 7.2 /home/timurkarimov/work/arch-pc/lab06/lab6-3.asm
%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

```

Рис. 4.12: Создание файла и написание программы

2. Создаем исполняемый файл и запускаем его (рис. 4.13)

```

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

```

Рис. 4.13: Выполнение программы

Изменяем текст программы для вычисления нового выражения (рис. 4.14)

```

; ---- Вычисление выражения
mov eax,4 ; EAX=5
mov ebx,6 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; 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. 4.14: Изменение программы

Создаем исполняемый файл и запускаем его (рис. 4.15)

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

Рис. 4.15: Выполнение программы

С помощью утилиты `touch` создаем файл `variant.asm`. Вводим в него текст программы для вычисления варианта (рис. 4.16)

```
%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 sprintf
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
[ Прочитано 25 строк ]
```

Рис. 4.16: Создание файла и написание программы

Создаем исполняемый файл и запускаем его (рис. 4.17)

```
timurkarimov@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
timurkarimov@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246817
Ваш вариант: 18
```

Рис. 4.17: Выполнение программы

Ответы на вопросы по программе: 1. За вывод сообщения “Ваш вариант” отвечают

строки кода:

`mov eax, rem`

`call sprint`

2. Инструкция `mov esx, x` выполняется для того чтобы положить адрес вводимой строки `x` в регистр `esx`. Инструкция `mov edx, 80` выполняется для записи длины вводимой строки в регистр `edx`. Инструкция `call sread` выполняется для вызова подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры.
3. Инструкция `call atoi` используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`.
4. За вычисление варианта отвечают строки: `xor edx, edx`

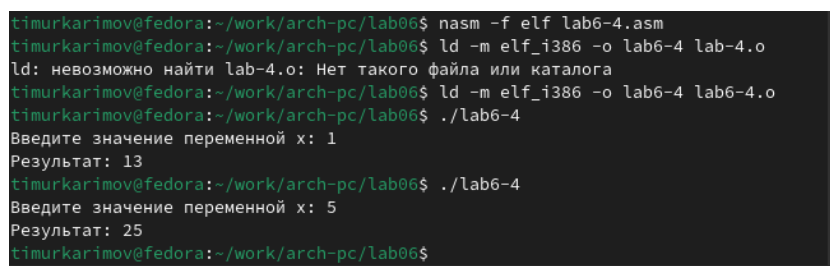
`mov ebx, 20`

`div ebx`

`inc edx`

5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`.
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1.
7. За вывод на экран результата вычислений отвечаю следующие строки: `mov eax, edx call iprintLF` #Выполнение самостоятельной работы

Создаем файл `lab6-4.asm` и записываем в него программу для вычисления выражения 18 варианта. (рис. 4.18)



```
timurkarimov@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
timurkarimov@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab-4.o
ld: невозможно найти lab-4.o: Нет такого файла или каталога
timurkarimov@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
timurkarimov@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 13
timurkarimov@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 5
Результат: 25
timurkarimov@fedora:~/work/arch-pc/lab06$
```

Рис. 4.18: Создание файла и написание программы

Создаем исполняемый файл и проверяем его работу для указанных значений (рис. 4.19)


```

GNU nano 7.2 /home/timurkarimov/work/arch-pc/lab06/lab6-4.asm
#include 'in_out.asm' ; подключение внешнего файла

SECTION .data ; секция инициированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss ; секция не инициированных данных
x: RESB 80 ; переменная, значение к-рой будем ввод>
; с клавиатуры, выделенный размер - 80 >

SECTION .text ; код программы
GLOBAL _start ; начало программы
_start: ; точка входа в программу

; ---- Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в>
call sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax,x ; вызов подпрограммы преобразования

```

Рис. 4.19: Выполнение программы

5 Выводы

При выполнении лабораторной работы помогла освоению арифметических инструкций языка ассемблера NASM.

Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.
2. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.