

Шаблон отчёта по лабораторной работе

Простейший вариант

плето мбамби

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	8
4	Выводы	12
	Список литературы	13

Список иллюстраций

3.1	Создайте каталог для работы	8
3.2	вставить текст hello.asm	9
3.3	Расширенный синтаксис командной строки NASM	9
3.4	Запуск исполняемого файла	9
3.5	копию файла	10
3.6	текст программы	10
3.7	lab5.asm	11
3.8	Скопируйте файлы	11

Список таблиц

1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM. # Задание

1. В каталоге `~/work/arch-pc/lab05` с помощью команды `cp` создайте копию файла `hello.asm` с именем `lab5.asm`
2. С помощью любого текстового редактора внесите изменения в текст программы в файле `lab5.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с вашими фамилией и именем.
3. Оттранслируйте полученный текст программы `lab5.asm` в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл.
4. Скопируйте файлы `hello.asm` и `lab5.asm` в Ваш локальный репозиторий в каталог `~/work/study/2022-2023/“Архитектура компьютера”/arch-pc/labs/lab05/`. Загрузите файлы на Github.

2 Теоретическое введение

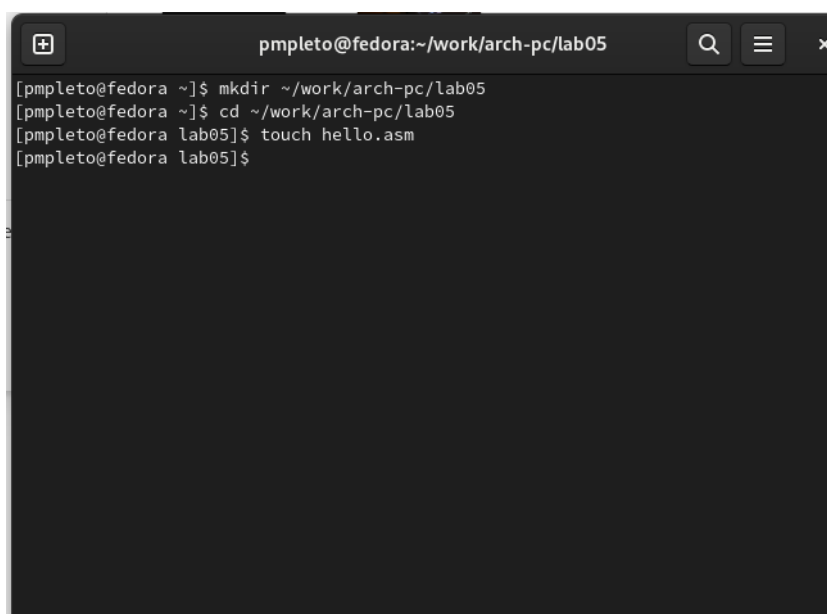
Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства (рис. 5.1). Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской (системной) плате. результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры. Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах написанных на ассемблере используют регистры в качестве операндов. Практически все команды представляют собой преобразование данных хранящихся в регистрах процессора, это например пересылка данных между регистрами или между регистрами и памятью, преобразование (арифметические или логические операции) данных хранящихся в регистрах. Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита. В качестве примера приведем названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ):

- RAX, RCX, RDX, RBX, RSI, RDI — 64-битные
- EAX, ECX, EDX, EBX, ESI, EDI — 32-битные
- AX, CX, DX, BX, SI, DI — 16-битные
- AH, AL, CH, CL, DH, DL, BH,

BL — 8-битные (половинки 16-битных регистров). Например, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX.

3 Выполнение лабораторной работы

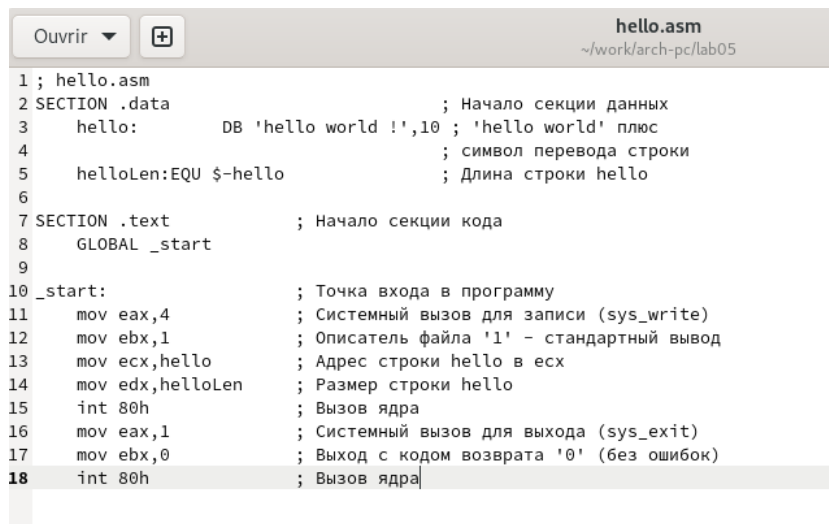
1. Создайте каталог для работы с программами на языке ассемблера NASM.
(рис. 3.1)

A terminal window with a dark background. The title bar shows the user 'pmp1eto@fedora' and the current directory '~/work/arch-pc/lab05'. The terminal contains the following commands and their outputs:

```
[pmp1eto@fedora ~]$ mkdir ~/work/arch-pc/lab05
[pmp1eto@fedora ~]$ cd ~/work/arch-pc/lab05
[pmp1eto@fedora lab05]$ touch hello.asm
[pmp1eto@fedora lab05]$
```

Рис. 3.1: Создайте каталог для работы

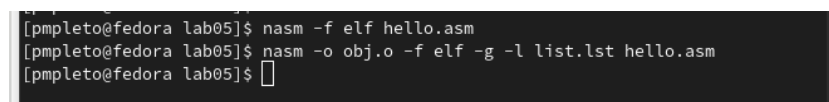
2. файл с помощью любого текстового редактора, например, gedit с помощью команды gedit hello.asm. (рис. 3.2)



```
1 ; hello.asm
2 SECTION .data                ; Начало секции данных
3     hello:                   DB 'hello world !',10 ; 'hello world' плюс
4                               ; символ перевода строки
5     helloLen:EQU $-hello      ; Длина строки hello
6
7 SECTION .text                ; Начало секции кода
8     GLOBAL _start
9
10 _start:                     ; Точка входа в программу
11     mov eax,4                ; Системный вызов для записи (sys_write)
12     mov ebx,1                ; Описатель файла '1' - стандартный вывод
13     mov ecx,hello            ; Адрес строки hello в есх
14     mov edx,helloLen         ; Размер строки hello
15     int 80h                 ; Вызов ядра
16     mov eax,1                ; Системный вызов для выхода (sys_exit)
17     mov ebx,0                ; Выход с кодом возврата '0' (без ошибок)
18     int 80h                 ; Вызов ядра
```

Рис. 3.2: вставить текст hello.asm

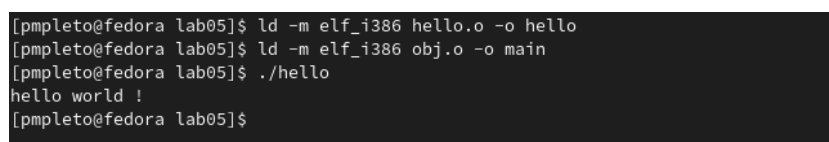
3. Транслятор NASMКомпоновщик LD. (рис. 3.3)



```
[pmp1eto@fedora lab05]$ nasm -f elf hello.asm
[pmp1eto@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[pmp1eto@fedora lab05]$
```

Рис. 3.3: Расширенный синтаксис командной строки NASM

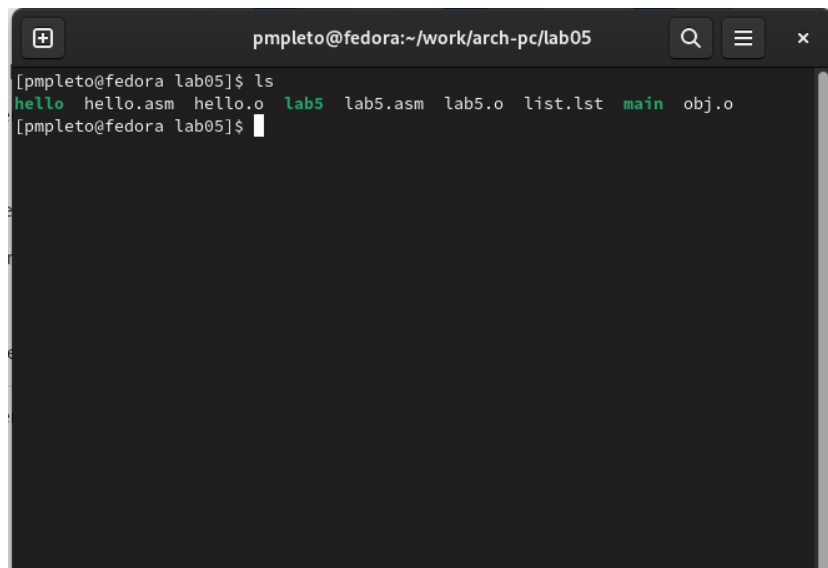
4. Компоновщик LD. (рис. 3.4)



```
[pmp1eto@fedora lab05]$ ld -m elf_i386 hello.o -o hello
[pmp1eto@fedora lab05]$ ld -m elf_i386 obj.o -o main
[pmp1eto@fedora lab05]$ ./hello
hello world !
[pmp1eto@fedora lab05]$
```

Рис. 3.4: Запуск исполняемого файла

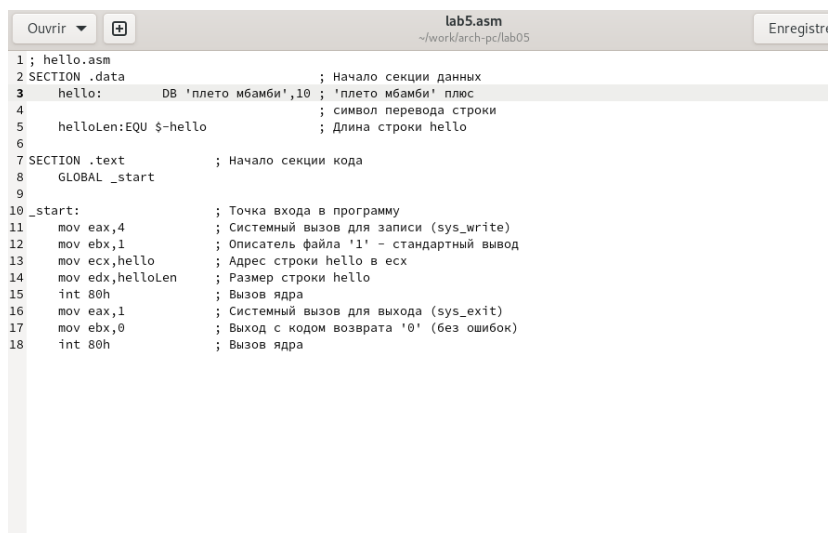
5. В каталоге ~/.work/arch-pc/lab05 с помощью команды cp создайте копию файла hello.asm с именем lab5.asm. (рис. 3.5)



```
pmp1eto@fedora lab05]$ ls
hello  hello.asm  hello.o  lab5  lab5.asm  lab5.o  list.lst  main  obj.o
pmp1eto@fedora lab05]$
```

Рис. 3.5: копию файла

6. С помощью любого текстового редактора внесите изменения в текст программы в файле lab5.asm так, чтобы вместо Hello world! на экран выводилась строка с вашими фамилией и именем. (рис. 3.6)



```
lab5.asm
~/work/arch-pc/lab05
Enregistrer

1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'плето мбамби',10 ; 'плето мбамби' плюс
4 ; символ перевода строки
5 helloLen:EQU $-hello ; Длина строки hello
6
7 SECTION .text ; Начало секции кода
8 GLOBAL _start
9
10 _start: ; Точка входа в программу
11 mov eax,4 ; Системный вызов для записи (sys_write)
12 mov ebx,1 ; Описатель файла '1' - стандартный вывод
13 mov ecx,hello ; Адрес строки hello в ecx
14 mov edx,helloLen ; Размер строки hello
15 int 80h ; Вызов ядра
16 mov eax,1 ; Системный вызов для выхода (sys_exit)
17 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
18 int 80h ; Вызов ядра
```

Рис. 3.6: текст программы

7. Оттранслируйте полученный текст программы lab5.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл. (рис. 3.7)

```

pmp1eto@fedora:~/work/arch-pc/lab05
[pmp1eto@fedora lab05]$ touch lab5.asm
[pmp1eto@fedora lab05]$ nasm -f elf lab5.asm
[pmp1eto@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst lab5.asm
[pmp1eto@fedora lab05]$ ld -m elf_i386 lab5.o -o lab5
[pmp1eto@fedora lab05]$ ld -m elf_i386 obj.o -o main
[pmp1eto@fedora lab05]$ ./lab5
плето мбамби
[pmp1eto@fedora lab05]$

```

Рис. 3.7: lab5.asm

8. Скопируйте файлы hello.asm и lab5.asm в Ваш локальный репозиторий в каталог ~/work/study/2022-2023/“Архитектура компьютера”/arch-rc/labs/lab05/,Загрузите файлы на Github. (рис .3.8)

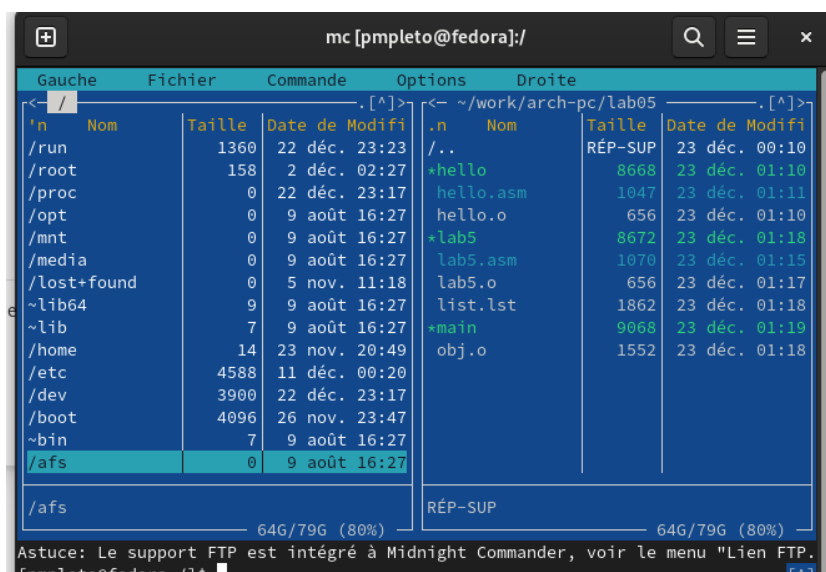


Рис. 3.8: Скопируйте файлы

4 Выводы

В ходе этой лабораторной работы я приобрел практический навык в освоении процедур компиляции и ассемблера программ, написанных на ассемблере NASM.

Список литературы

1. Расширенный ассемблер: NASM