

Отчёт по лабораторной работе №4

Дисциплина: Архитектура компьютеров.

Ворожейкин Владимир Вячеславович

Содержание

Цель работы.....	1
Задание.....	1
Теоретическое введение.....	1
Выполнение лабораторной работы.....	3
Создание программы Hello world!.....	3
Работа с транслятором NASM.....	3
Работа с расширенным синтаксисом командной строки NASM.....	3
Работа с компоновщиком LD.....	3
Запуск исполняемого файла.....	3
Выполнение заданий для самостоятельной работы.....	3
Bookmark not defined.	Error!
Выводы.....	5
Список литературы.....	5

Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

Задание

1. Создание программы Hello world!
2. Работа с транслятором NASM
3. Работа с расширенным синтаксисом командной строки NASM
4. Работа с компоновщиком LD
5. Запуск исполняемого файла

6. Выполнение заданий для самостоятельной работы.

Теоретическое введение

Основные функциональные элементы ЭВМ (электронно-вычислительные машины) работают совместно для выполнения операций обработки информации. Ниже описаны несколько ключевых элементов и их работы.

1. Центральный процессор (ЦП) - это “мозг” компьютера. Он выполняет основные вычисления, управляет и контролирует операции в ЭВМ. ЦП состоит из нескольких частей:

- Арифметико-логическое устройство (АЛУ) - выполняет арифметические и логические операции, например, сложение, вычитание, умножение, деление и сравнение.
- Устройство управления - принимает команды и данные, декодирует их и управляет внутренними ресурсами ЦП, такими как регистры и память.
- Регистры - это небольшие хранилища данных в ЦП. Они могут содержать команды, данные и результаты вычислений на протяжении операций.

2. Оперативная память (ОЗУ) - это тип хранилища данных, используемый для временного хранения команд и данных, которые ЦП активно использует. ОЗУ обеспечивает быстрый доступ к данным, и они могут быть прочитаны или записаны через различные адресные линии.

3. Устройства ввода/вывода (В/В) - это устройства, используемые для передачи информации между ЭВМ и внешними устройствами. Примеры устройств В/В включают клавиатуру, мышь, монитор, дисковод, принтер и т.д. Они обеспечивают ввод данных в ЭВМ и вывод результатов.

4. Арифметический логический блок (ALU) - отвечает за выполнение арифметических и логических операций, таких как сложение, вычитание, умножение, деление, сравнение и логические операции (НЕ, ИЛИ, И, XOR и т.д.).

5. Память для хранения данных (например, жесткий диск) - это основное устройство для долговременного хранения данных в ЭВМ. Он используется для сохранения операционной системы, программного обеспечения, файлов пользователя и других данных.

6. Шины - это набор проводов, которые используются для передачи данных, команд и сигналов между различными компонентами ЭВМ, такими как ЦП, ОЗУ и В/В устройства. Все эти элементы работают совместно, обмениваясь информацией и выполняя операции в соответствии с программами и командами, заданными пользователем. Они обеспечивают эффективное выполнение задач и обработку информации в ЭВМ.

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64.

Выполнение лабораторной работы

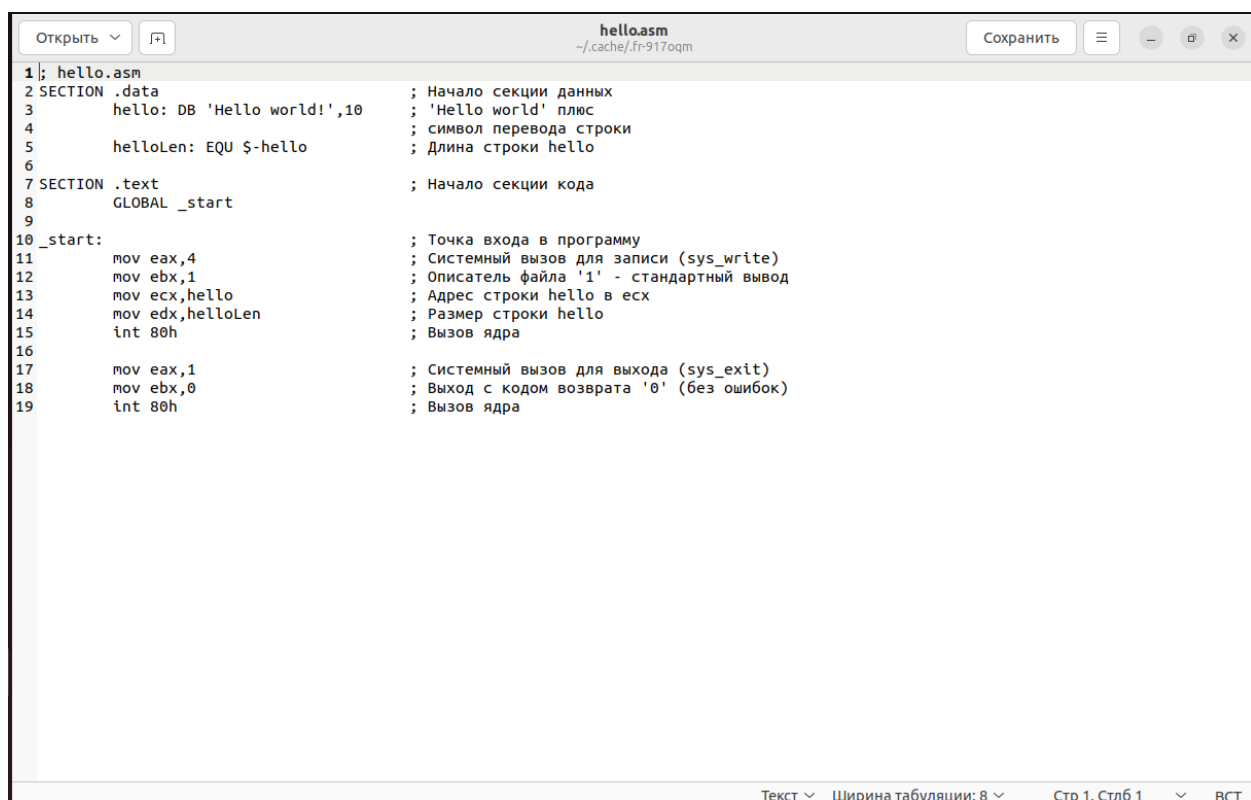
Создание программы Hello world!

С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать (рис. 1), создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью `touch`.

```
ubuntu@ubuntu-VirtualBox:~$ cd ~/work/study/2023-2024/Computer_architecture/arch-pc/labs/lab04
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer_architecture/arch-pc/labs/lab04$ touch hello.asm
```

Перемещение между директориями

Заполняю файл, вставляя в него программу для вывода “Hello world!” (рис. 2).



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

Заполнение файла

Работа с транслятором NASM.

Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду `nasm -f elf hello.asm`, ключ `-f` указывает транслятору `nasm`, что требуется создать бинарный файл в формате ELF (рис. 3).

```
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer_architecture/arch-pc/labs/lab04$ nasm -f elf hello.asm
```

Компиляция текста программы

```
ubuntubest@ubuntubest-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  presentation  report
```

Далее проверяю правильность выполнения команды с помощью ls (рис.4).

Проверка правильности выполнения

Работа с расширенным синтаксисом командной строки NASM.

Ввожу команду, которая скомпилирует файл hello.asm в файл obj.o, используя ключ -o который задает имя объектному файлу, так же в файл будут включены символы для отладки (ключ -g), с помощью ключа -l будет создан файл list.lst (рис. 5). Далее проверяю с помощью ls правильность выполнения команды.

```
ubuntubest@ubuntubest-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst
ubuntubest@ubuntubest-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$
```

Компиляция текста программы

Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. 6). Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью ls правильность выполнения команды (рис.6)

```
ubuntubest@ubuntubest-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
ubuntubest@ubuntubest-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  lab04.asm  list.lst  main  obj.o  presentation  report
```

Компиляция текста программы

Выполняю следующую команду (рис. 7). Исполняемый файл будет иметь имя main, т.к. после ключа -o было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o. С помощью ls проверяю правильность выполнения.

```
ubuntubest@ubuntubest-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 obj.o -o main
ubuntubest@ubuntubest-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  lab04.asm  list.lst  main  obj.o  presentation  report
ubuntubest@ubuntubest-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$
```

Передача объектного файла на обработку компоновщику

Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. 8).

```
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ ./hello
Hello world!
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$
```

Запуск исполняемого файла

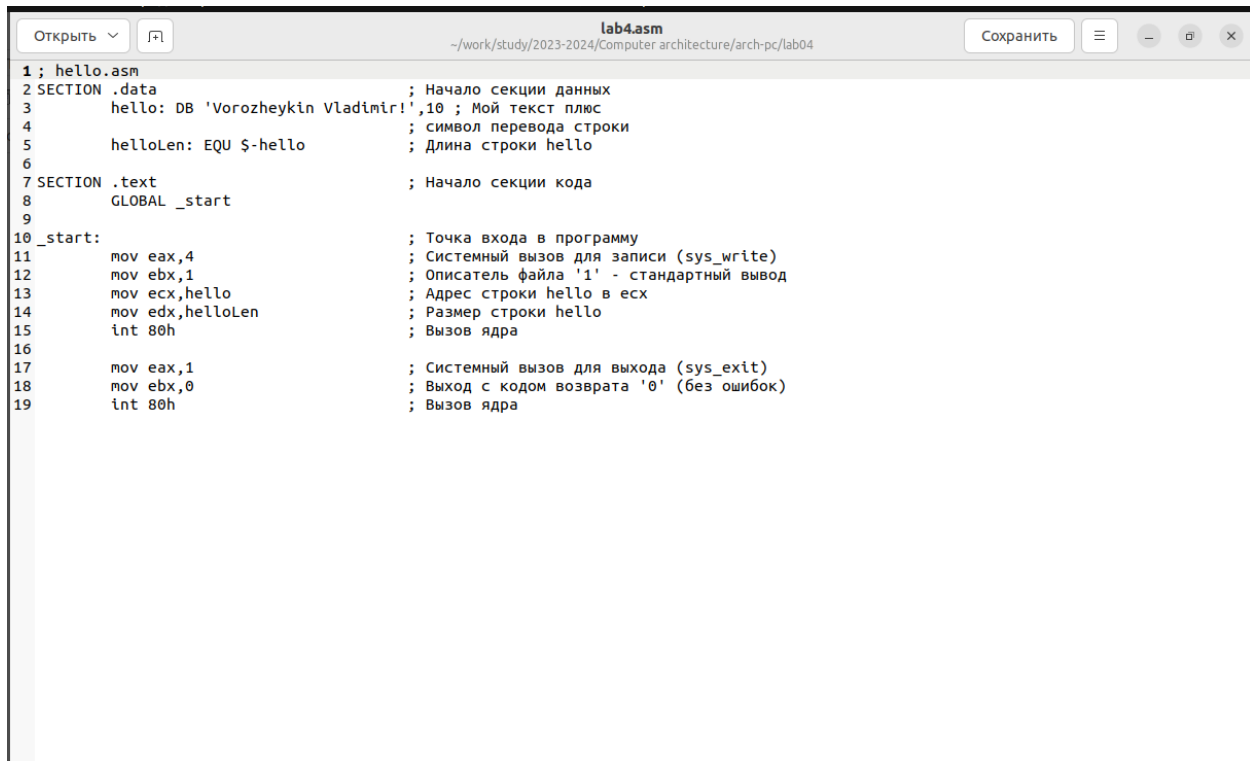
Выполнение заданий для самостоятельной работы.

С помощью `ср` создаю в текущем каталоге копию файла `hello.asm` с именем `lab4.asm` (рис. 9).

```
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ cp hello.asm lab04.asm
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$
```

Создание копии файла

С помощью текстового редактора открываю файл `lab4.asm` и вношу изменения в программу, чтобы она выводила мои имя и фамилию. (рис. 10).



```
lab4.asm
~/work/study/2023-2024/Computer architecture/arch-pc/lab04
Сохранить

1 ; hello.asm
2 SECTION .data                ; Начало секции данных
3     hello: DB 'Vorozheykin Vladimir!',10 ; Мой текст 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
17     mov eax,1                ; Системный вызов для выхода (sys_exit)
18     mov ebx,0                ; Выход с кодом возврата '0' (без ошибок)
19     int 80h                  ; Вызов ядра
```

Изменение программы

Компилирую текст программы в объектный файл (рис. 11). Проверяю с помощью `ls`, что файл `lab4.o` создан.

```
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ nasm -f elf lab04.asm
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ ls
hello hello.asm hello.o lab04.asm lab04.o list.lst main obj.o presentation report
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$
```

Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. 12). С помощью ls проверяю правильность выполнения.

```
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 lab04.o -o lab04
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ ls
hello hello.asm hello.o lab04 lab04.asm lab04.o list.lst main obj.o presentation report
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$
```

Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, на экран выводятся мои имя и фамилия (рис. 13).

Запуск исполняемого файла

```
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ ./lab4
Vorozheykin Vladimir!
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$
```

С помощью команды git add . добавляю файлы на GitHub (рис. 14).

```
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ git add .
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$
```

Добавление файлов на GitHub

Отправляю файлы на сервер с помощью команды git push (рис. 15).

```
ubuntu@ubuntu-VirtualBox:~/work/study/2023-2024/Computer architecture/arch-pc/labs/lab04$ git push
Username for 'https://github.com': vvvorozheykin
Password for 'https://vvvorozheykin@github.com':
Перечисление объектов: 22, готово.
Подсчет объектов: 100% (22/22), готово.
Сжатие объектов: 100% (17/17), готово.
Запись объектов: 100% (17/17), 47.95 Киб | 23.97 Миб/с, готово.
Всего 17 (изменений 7), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (7/7), completed with 2 local objects.
To https://github.com/vvvorozheykin/study_2023-2024_arch-pc.git
72291ff..5bb078f master -> master
```

Отправка файлов

Выводы

При выполнении этой лабораторной работы я освоил способы компиляции и сборки программ, написанных на ассемблере NASM.

Список литературы

1.

https://esystem.rudn.ru/pluginfile.php/1584628/mod_resource/content/1/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%965.pdf