

Лабораторная работа №4

Создание и процесс обработки программ на языке ассемблера NASM

Селиванов Вячеслав Алексеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Создание программы Hello world!	8
4.2	Работа с транслятором NASM.	9
4.3	Работа с компоновщиком LD.	10
4.4	Запуск исполняемого файла.	11
4.5	Выполнение заданий для самостоятельной работы.	11
5	Выводы	15
	Список литературы	16

Список иллюстраций

4.1	Перемещение по директориям и создание файла	8
4.2	Файл hello.asm	8
4.3	Программа	9
4.4	Создание бинарного файла	9
4.5	Создание бинарного файла	10
4.6	Создание файла листинга	10
4.7	Создание бинарного файла	10
4.8	Получение исполняемого файла	10
4.9	Значение main	11
4.10	Проверка программы	11
4.11	Копирование файла hello.asm	11
4.12	Изменение программы	12
4.13	Создание объектного файла	12
4.14	Получение исполняемого файла	12
4.15	Проверка программы	13
4.16	Добавление файлов в лабораторную работу	13
4.17	Отправление работы на github	14

Список таблиц

1 Цель работы

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

2 Задание

Создание программы Hello world!

Работа с транслятором NASM

Работа с расширенным синтаксисом командной строки NASM

Работа с компоновщиком LD

Запуск исполняемого файла

Выполнение заданий для самостоятельной работы.

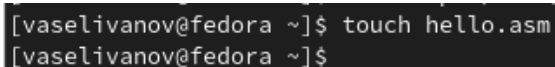
3 Теоретическое введение

Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства (рис. 4.1). 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. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора (ЦП) входят следующие устройства: • арифметико-логическое устройство (АЛУ) — выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти; • устройство управления (УУ) — обеспечивает управление и контроль всех устройств компьютера; • регистры — сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры.

4 Выполнение лабораторной работы

4.1 Создание программы Hello world!

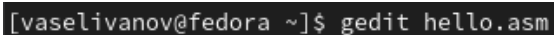
Создаю пустой текстовый файл hello.asm(рис. 4.1).



```
[vaselivanov@fedora ~]$ touch hello.asm  
[vaselivanov@fedora ~]$
```

Рис. 4.1: Перемещение по директориям и создание файла

Открываю файл в текстовом редакторе gedit(рис. 4.2).



```
[vaselivanov@fedora ~]$ gedit hello.asm
```

Рис. 4.2: Файл hello.asm

Заполняю файл, заполняю программу для вывода Hello world (рис. ??).

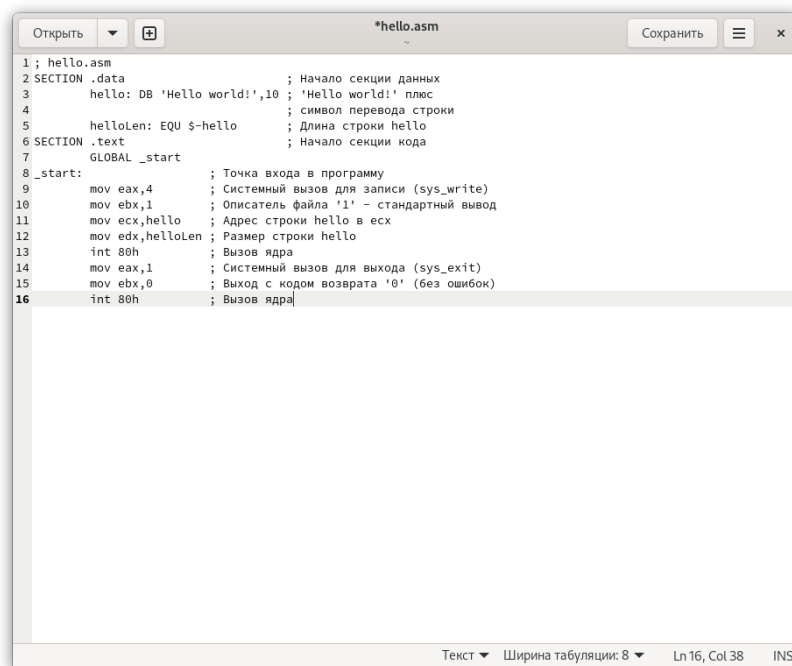


Рис. 4.3: Программа

4.2 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 и с помощью утилиты `ls` проверяю создан ли файл `hello.o` (рис. 4.4).

```

[vaselivanov@fedora ~]$ nasm -f elf hello.asm
[vaselivanov@fedora ~]$

```

Рис. 4.4: Создание бинарного файла

С помощью утилиты `ls` проверяю создан ли файл `hello.o` (рис. 4.5).

```
[vaselivanov@fedora ~]$ ls
hello.asm  study_2023-2024_arh-pc  Видео  Загрузки  Музыка  'Рабочий стол'
hello.o    work                     Документы  Изображения  Общедоступные  Шаблоны
```

Рис. 4.5: Создание бинарного файла

##Работа с расширенным синтаксисом командной строки NASM. Ввожу команду, которая скомпилирует файл hello.asm в файл obj.o, при этом в файл будут включены символы для отладки (ключ -g), также с помощью ключа -l будет создан файл листинга list.lst. (рис. 4.6).

```
hello.o    work                     Документы  Изображения  Общедоступные  Шаблоны
[vaselivanov@fedora ~]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[vaselivanov@fedora ~]$
```

Рис. 4.6: Создание файла листинга

С помощью утилиты ls проверяю создан ли файл obj.o и list.lst (рис. 4.7).

```
[vaselivanov@fedora ~]$ ls
hello.asm  list.lst  study_2023-2024_arh-pc  Видео  Загрузки  Музыка  'Рабочий стол'
hello.o    obj.o     work                     Документы  Изображения  Общедоступные  Шаблоны
```

Рис. 4.7: Создание бинарного файла

4.3 Работа с компоновщиком LD.

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello. Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды. (рис. 4.8).

```
[vaselivanov@fedora ~]$ ls
hello      hello.o  obj.o      work  Документы  Изображения  Общедоступные  Шаблоны
hello.asm  list.lst  study_2023-2024_arh-pc  Видео  Загрузки  Музыка  'Рабочий стол'
```

Рис. 4.8: Получение исполняемого файла

Выполняю следующую команду 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. (рис. 4.9).

```
[vaselivanov@fedora ~]$ ls
hello      hello.o    main      study_2023-2024_arh-pc  Видео  Загрузки  Музыка
hello.asm  list.lst  obj.o     work                   Документы  Изображения  Общедоступны
```

Рис. 4.9: Значение main

4.4 Запуск исполняемого файла.

Запускаю созданный файл, чтобы проверить программу. (рис. 4.10).

```
[vaselivanov@fedora ~]$ ./hello
Hello world!
```

Рис. 4.10: Проверка программы

4.5 Выполнение заданий для самостоятельной работы.

Создаю копию файла, называю его lab04.asm. (рис. 4.11)

```
[vaselivanov@fedora ~]$ cp hello.asm lab04.asm
```

Рис. 4.11: Копирование файла hello.asm

С помощью текстового редактора gedit открываю файл lab04.asm и меняю программу так, чтобы она выводила мои имя и фамилию.(рис. 4.12)

```

1 ; lab04.asm
2 SECTION .data
3     lab04: DB 'SelivanovVyacheslav',10
4
5     lab04Len: EQU $-lab04          ; Длина строки lab04
6 SECTION .text                      ; Начало секции кода
7     GLOBAL _start
8 _start:                            ; Точка входа в программу
9     mov eax,4                      ; Системный вызов для записи (sys_write)
10    mov ebx,1                      ; Описатель файла '1' - стандартный вывод
11    mov ecx,lab04                  ; Адрес строки lab04 в ecx
12    mov edx,lab04Len               ; lab04Len - константа, а не переменная
13    int 80h                       ; Вызов ядра
14    mov eax,1                      ; Системный вызов для выхода (sys_exit)
15    mov ebx,0                      ; Выход с кодом возврата '0' (без ошибок)
16    int 80h                       ; Вызов ядра

```

Рис. 4.12: Изменение программы

Компилирую текст программы в объектный файл.Проверяю с помощью утилиты ls, что файл lab04.o создан.(рис. 4.11)

```

[vaselivanov@fedora lab04]$ nasm -f elf lab04.asm
[vaselivanov@fedora lab04]$ ls
hello hello.asm hello.o lab04.asm lab04.o list.lst main obj.o
[vaselivanov@fedora lab04]$

```

Рис. 4.13: Создание объектного файла

Передаю объектный файл lab04.o на обработку компоновщику LD, чтобы получить исполняемый файл lab04.(рис. 4.14)

```

[vaselivanov@fedora lab04]$ ld -m elf_i386 lab04.o -o lab04
[vaselivanov@fedora lab04]$ ls
hello hello.asm hello.o lab04 lab04.asm lab04.o list.lst main
[vaselivanov@fedora lab04]$

```

Рис. 4.14: Получение исполняемого файла

Запускаю исполняемый файл lab04, на экран действительно выводятся мои имя и фамилия.(рис. 4.15)

```
[vaselivanov@fedora lab04]$ ./lab04
SelivanovVyacheslav
[vaselivanov@fedora lab04]$
```

Рис. 4.15: Проверка программы

Скидываю файлы hello.asm и lab04.asm в каталог 4 лабораторный работы.(рис. 4.16)

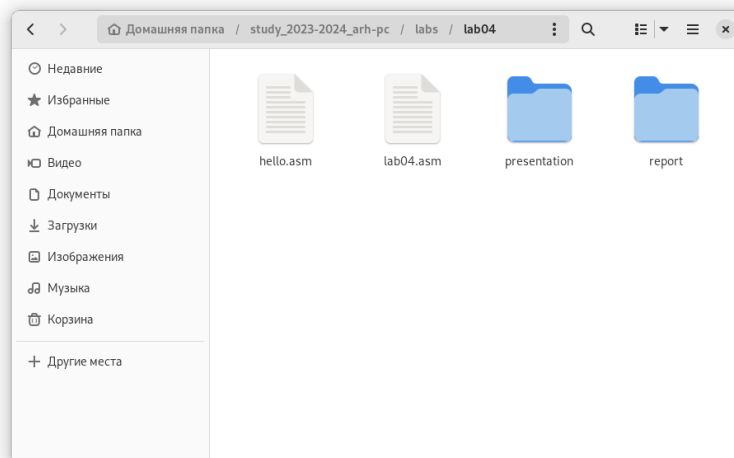


Рис. 4.16: Добавление файлов в лабораторную работу

Отправление всех изменений на github. (рис. 4.17)

```
[vaselivanov@fedora lab04]$ git add .
[vaselivanov@fedora lab04]$ git commit -am 'feat(main): add files lab04'
[master fdde434] feat(main): add files lab04
19 files changed, 131 insertions(+), 33 deletions(-)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab04.asm
create mode 100644 labs/lab04/report/image/1.png
create mode 100644 labs/lab04/report/image/10.png
create mode 100644 labs/lab04/report/image/11.png
create mode 100644 labs/lab04/report/image/12.png
create mode 100644 labs/lab04/report/image/13.png
create mode 100644 labs/lab04/report/image/14.png
create mode 100644 labs/lab04/report/image/15.png
create mode 100644 labs/lab04/report/image/2.png
create mode 100644 labs/lab04/report/image/3.png
create mode 100644 labs/lab04/report/image/4.png
create mode 100644 labs/lab04/report/image/5.png
create mode 100644 labs/lab04/report/image/6.png
create mode 100644 labs/lab04/report/image/7.png
create mode 100644 labs/lab04/report/image/8.png
create mode 100644 labs/lab04/report/image/9.png
delete mode 100644 labs/lab04/report/image/placeimg_800_600_tech.jpg
[vaselivanov@fedora lab04]$ git push
```

Рис. 4.17: Отправление работы на github

5 Выводы

При выполнении данной лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

Список литературы

(1Архитектура ЭВМ)