

Отчёт по лабораторной работе №4

Дисциплина: архитектура компьютера

Маньковская Дарья Станиславовна

Содержание

Список иллюстраций

1 Цель работы

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

2 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.

3 Теоретическое введение

Основными функциональными элементами любой ЭВМ являются центральный процессор, па-
логическое устройство (АЛУ) — выполняет логические и арифметические действия, не-
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


Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ — это быстродействующее энергозависимое запоминающее устройство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий момент. ОЗУ состоит из одинаковых пронумерованных ячеек памяти. Номер ячейки памяти — это адрес хранящихся в ней данных. Периферийные устройства в составе ЭВМ: - устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных. - устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой.

В основе вычислительного процесса ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы.

Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. Он заключается в следующем: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её дешифрация; 3. выполнение команды; 4. переход к следующей команде. 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.

4 Выполнение лабораторной работы

С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать (рис. 1).



```
dsmanjkovskaya@dk3n37 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Рис. 1. Перемещение между директориями

Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch` (рис. 2).



```
dsmanjkovskaya@dk3n37 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ touch hello.asm
```

Рис. 2: Создание пустого файла

Открываю созданный файл в текстовом редакторе `mousepad` (рис. 3).

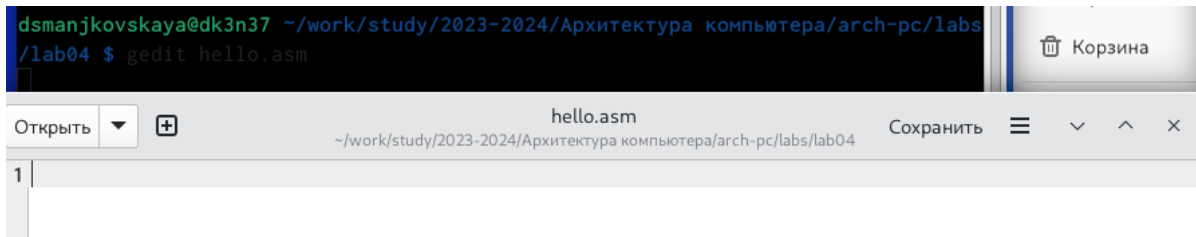


Рис. 3: Открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello word!” (рис. 4).

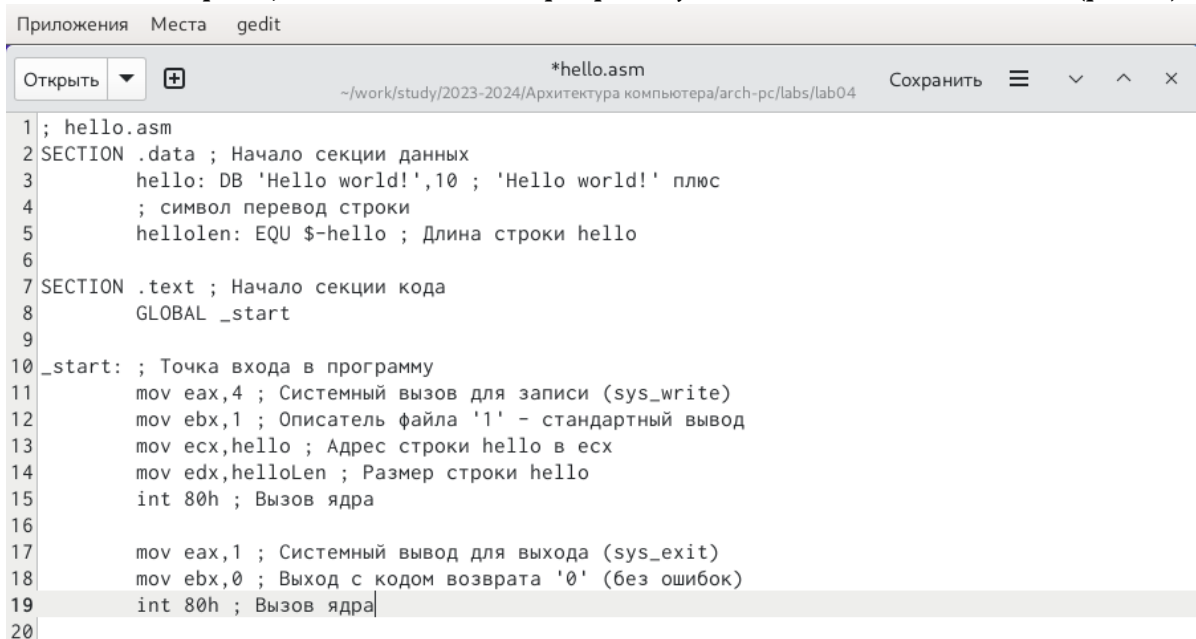


Рис. 4. Заполнение файла

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. 5). Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “hello.o”.

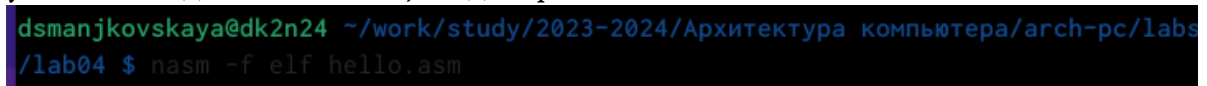


Рис. 5. Компиляция текста программы

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. 6). Далее проверяю с помощью утилиты

ls правильность выполнения команды.

```
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ls
hello.asm hello.o presentation report
```

Рис. 6. Компиляция текста программы

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. 7). Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды.

```
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ls
hello.asm hello.o list.lst obj.o presentation report
```

Рис. 7. Передача объектного файла на обработку компоновщику

Выполняю следующую команду (рис. 8). 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

```
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ld -m elf_i386 hello.o -o hello
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ls
hello hello.asm hello.o list.lst obj.o presentation report
```

Рис. 8. Передача объектного файла на обработку компоновщику

Запускаю на выполнение созданный исполняемый файл hello (рис. 9).

```
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ld -m elf_i386 obj.o -o main
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ls
hello hello.asm hello.o list.lst main obj.o presentation report
```

Рис. 9. Запуск исполняемого файла

С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab5.asm (рис. 10).

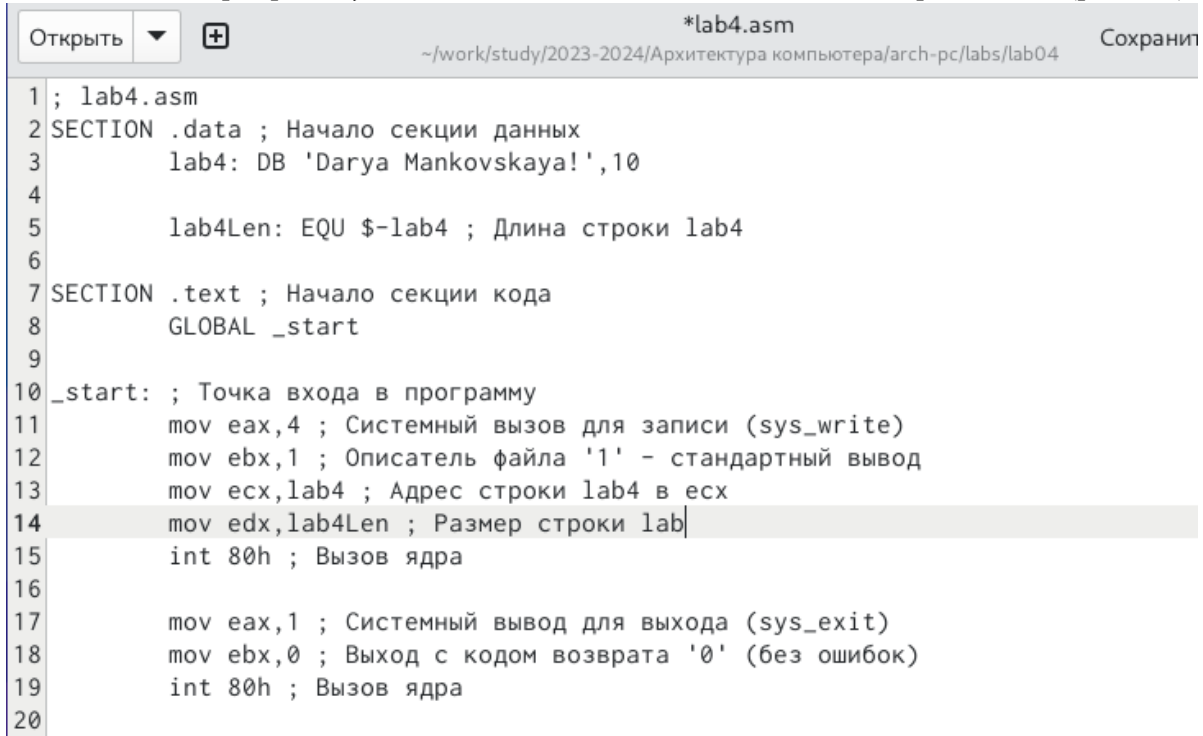
```

dsmanjovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ./hello
Hello world!

```

Рис. 10. Создание копии файла

С помощью текстового редактора mousepad открываю файл lab5.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис. 11).



```

Открыть ▼ + *lab4.asm ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 Сохранить
1 ; lab4.asm
2 SECTION .data ; Начало секции данных
3     lab4: DB 'Darya Mankovskaya!',10
4
5     lab4Len: EQU $-lab4 ; Длина строки lab4
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,lab4 ; Адрес строки lab4 в ecx
14     mov edx,lab4Len ; Размер строки lab4
15     int 80h ; Вызов ядра
16
17     mov eax,1 ; Системный вывод для выхода (sys_exit)
18     mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
19     int 80h ; Вызов ядра
20

```

Рис. 11. Изменение программы

Компилирую текст программы в объектный файл (рис. 12). Проверяю с помощью утилиты ls, что файл lab5.o создан.

```

dsmanjovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ nasm -f elf lab4.asm
dsmanjovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ls
hello      hello.o    lab4.o     list.lst   obj.o      report
hello.asm  lab4.asm  lab5.asm  main       presentation

```

Рис. 12. Компиляция текста программы

Передаю объектный файл lab5.o на обработку компоновщику LD, чтобы получить исполняемый файл lab5 (рис. 13).

```

dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ld -m elf_i386 lab4.o -o lab4
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ls
hello      hello.o  lab4.asm  lab5.asm  main      presentation
hello.asm  lab4     lab4.o    list.lst  obj.o     report

```

Рис. 13. Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab5, на экран действительно выводятся мои имя и фамилия (рис. 14).

```

dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ./lab4
Darya Mankovskaya

```

Рис. 14. Запуск исполняемого файла

К сожалению, я начала работу не в том каталоге, поэтому создаю другую директорию lab05 с помощью mkdir, прописывая полный путь к каталогу, в котором хочу создать эту директорию. Далее копирую из текущего каталога файлы, созданные в процессе выполнения лабораторной работы, с помощью утилиты cp, указывая вместо имени файла символ *, чтобы скопировать все файлы. Команда проигнорирует директории в этом каталоге, т. к. не указан ключ -r, это мне и нужно (рис. 15). Проверяю с помощью утилиты ls правильность выполнения команды.

```

dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ mkdir ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab04
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ cp * ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab04
cp: не указан -r; пропускается каталог 'presentation'
cp: не указан -r; пропускается каталог 'report'
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ls ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab04
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o

```

Рис. 15. Создании копии файлов в новом каталоге

Удаляю лишние файлы в текущем каталоге с помощью утилиты rm, ведь копии файлов остались в другой директории (рис. 16).


```

dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ rm hello hello.o lab4 lab4.o list.lst main obj.o
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ ls
hello.asm lab4.asm presentation report

```

Рис. 16. Удаление лишних файлов в текущем каталоге

С помощью команд `git add .` и `git commit` добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №5 (рис. 17).

```

dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ git add .
dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ git commit -m 'Add fales for lab04'
[master 07f2548] Add fales for lab04
 2 files changed, 40 insertions(+)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab4.asm

```

Рис. 17. Добавление файлов на GitHub

Отправляю файлы на сервер с помощью команды `git push` (рис. 18).

```

dsmanjkovskaya@dk2n24 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs
/lab04 $ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.05 КиБ | 1.05 МиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:derymank/study_2023-2024_arch-pc.git
 94861bf..07f2548 master -> master

```

Рис. 18. Отправка файл

5 Выводы

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

6 Список литературы

Архитектура ЭВМ