

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

дисциплина: Архитектура компьютера

Студент: Трофимов Владислав Алексеевич

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

Список иллюстраций	3
Список таблиц	4
1 Цель работы	5
2 Задание	6
3 Теоретическое введение	7
4 Выполнение лабораторной работы	10
4.1 Программа Hello world!	10
4.2 Транслятор NASM	11
4.3 Расширенный синтаксис командной строки NASM	11
4.4 Компоновщик LD	12
4.5 Запуск исполняемого файла	12
4.6 Задания для самостоятельной работы	13
5 Выводы	15
6 Список литературы	15

Список иллюстраций

- Рис. 4.1: Создание рабочей директории
- Рис. 4.2: Создание .asm файла
- Рис. 4.3: Редактирование файла
- Рис. 4.4: Компиляция программы
- Рис. 4.5: Возможности синтаксиса NASM
- Рис. 4.6: Отправка файла компоновщику
- Рис. 4.7: Создание исполняемого файла
- Рис. 4.8: Запуск программы
- Рис. 4.9: Создание копии
- Рис. 4.10: Редактирование копии
- Рис. 4.11: Проверка работоспособности скомпонованной программы
- Рис. 4.12: Отправка файлов в локальный репозиторий
- Рис. 4.13: Загрузка изменений

Список таблиц

1 Цель работы

Цель лабораторной работы - освоение 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 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. 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.

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.

В состав ЭВМ также входят периферийные устройства, которые можно разделить на:

- устройства внешней памяти, которые предназначены для длительного хранения больших объёмов данных (жёсткие диски, твердотельные накопители, магнитные ленты);
- 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. Программа состоит из машинных команд, которые указывают, какие операции и над какими данными (или операндами), в какой последовательности необходимо выполнить.

Набор машинных команд определяется устройством конкретного процессора. 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.

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что

позволяет получить к ним более полный доступ, нежели в языках высокого уровня, таких как C/C++, Perl, Python и пр. Заметим, что получить полный доступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру операционной системы. Именно на этом уровне и работают программы, написанные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора.

4 Выполнение лабораторной работы

4.1 Программа Hello world!

В домашней директории создаю каталог, в котором буду хранить файлы для текущей лабораторной работы, и перехожу в него. (рис. 4.1)

```
avtrofimov@tva-al:~$ mkdir -p ~/work/arh-pc/lab04  
avtrofimov@tva-al:~$ cd ~/work/arh-pc/lab04/  
avtrofimov@tva-al:~/work/arh-pc/lab04$ █
```

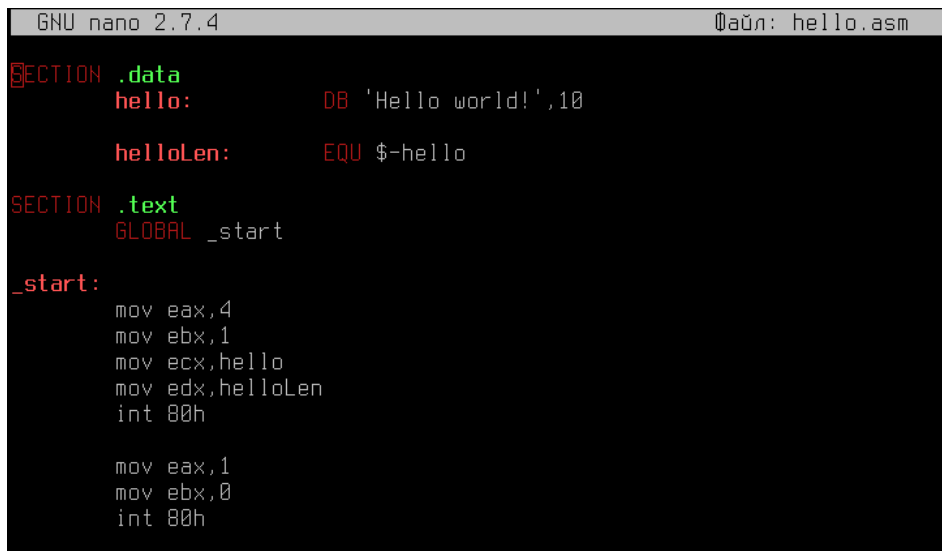
Рис. 4.1: Создание рабочей директории

Создаю в нем файл hello.asm, в котором буду писать программу на языке ассемблера. (рис. 4.2)

```
avtrofimov@tva-al:~/work/arh-pc/lab04$ touch hello.asm  
avtrofimov@tva-al:~/work/arh-pc/lab04$ nano hello.asm  
avtrofimov@tva-al:~/work/arh-pc/lab04$ █
```

Рис. 4.2: Создание .asm файла

С помощью редактора пишу программу в созданном файле. (рис. 4.3)



```
GNU nano 2.7.4                                Файл: hello.asm
SECTION .data
hello:      DB 'Hello world!',10

helloLen:   EQU $-hello

SECTION .text
GLOBAL _start

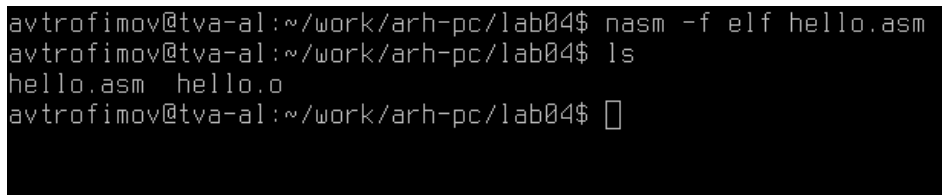
_start:
    mov eax,4
    mov ebx,1
    mov ecx,hello
    mov edx,helloLen
    int 80h

    mov eax,1
    mov ebx,0
    int 80h
```

Рис. 4.3: Редактирование файла

4.2 Транслятор NASM

Компилирую с помощью NASM свою программу. (рис. 4.4)

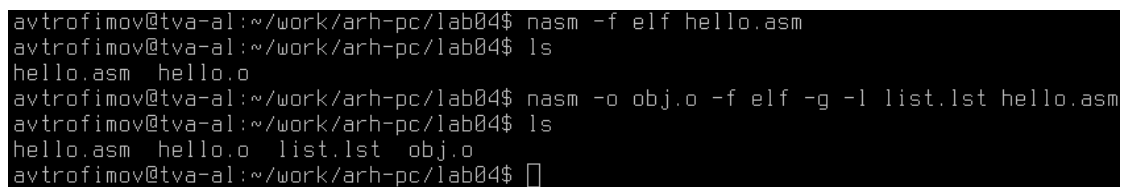


```
avtrofimov@tva-al:~/work/arh-pc/lab04$ nasm -f elf hello.asm
avtrofimov@tva-al:~/work/arh-pc/lab04$ ls
hello.asm  hello.o
avtrofimov@tva-al:~/work/arh-pc/lab04$
```

Рис. 4.4: Компиляция программы

4.3 Расширенный синтаксис командной строки NASM

Выполняю команду, указанную на (рис. 4.5), она скомпилировала исходный файл hello.asm в obj.o, расширение .o говорит о том, что файл - объектный, помимо него флаги -g -l подготовят файл отладки и листинга соответственно.



```
avtrofimov@tva-al:~/work/arh-pc/lab04$ nasm -f elf hello.asm
avtrofimov@tva-al:~/work/arh-pc/lab04$ ls
hello.asm  hello.o
avtrofimov@tva-al:~/work/arh-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
avtrofimov@tva-al:~/work/arh-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
avtrofimov@tva-al:~/work/arh-pc/lab04$
```

Рис. 4.5: Возможности синтаксиса NASM

4.4 Компоновщик LD

Затем мне необходимо передать объектный файл компоновщику, делаю это с помощью команды `ld`. (рис. 4.6)

```
avtrofimov@tva-al:~/work/arh-pc/lab04$ ld -m elf_i386 hello.o -o hello
avtrofimov@tva-al:~/work/arh-pc/lab04$ ls
hello hello.asm hello.o list.lst obj.o
avtrofimov@tva-al:~/work/arh-pc/lab04$
```

Рис. 4.6: Отправка файла компоновщику

Выполняю следующую команду ..., результатом исполнения команды будет созданный файл `main`, скомпонованный из объектного файла `obj.o`. (рис. 4.7)

```
avtrofimov@tva-al:~/work/arh-pc/lab04$ ld -m elf_i386 obj.o -o main
avtrofimov@tva-al:~/work/arh-pc/lab04$ ls
hello hello.asm hello.o list.lst main obj.o
avtrofimov@tva-al:~/work/arh-pc/lab04$
```

Рис. 4.7: Создание исполняемого файла

4.5 Запуск исполняемого файла

Запускаю исполняемый файл из текущего каталога. (рис. 4.8)

```
avtrofimov@tva-al:~/work/arh-pc/lab04$ ./hello
Hello world!
avtrofimov@tva-al:~/work/arh-pc/lab04$
```

Рис. 4.8: Запуск программы

4.6 Задания для самостоятельной работы

Создаю копию файла для последующей работы с ней. (рис. 4.9)

```
avtrofimov@tva-al:~/work/arh-pc/lab04$ cp hello.asm lab4.asm
avtrofimov@tva-al:~/work/arh-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
avtrofimov@tva-al:~/work/arh-pc/lab04$
```

Рис. 4.9: Создание копии

Редактирую копию файла, заменив текст на свое имя и фамилию. (рис. 4.10)

```
GNU nano 2.7.4
SECTION .data
    hello:      DB 'Trofimov Vladislav',10

    helloLen:   EQU $-hello

SECTION .text
    GLOBAL _start

_start:
    mov eax,4
    mov ebx,1
    mov ecx,hello
    mov edx,helloLen
    int 80h

    mov eax,1
    mov ebx,0
    int 80h
```

Рис. 4.10: Редактирование копии

Транслирую копию файла в объектный файл, компоную и запускаю. (рис. 4.11)

```
avtrofimov@tva-al:~/work/arh-pc/lab04$ nasm -f elf lab4.asm
avtrofimov@tva-al:~/work/arh-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
avtrofimov@tva-al:~/work/arh-pc/lab04$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
avtrofimov@tva-al:~/work/arh-pc/lab04$ ./lab4
Trofimov Vladislav
avtrofimov@tva-al:~/work/arh-pc/lab04$
```

Рис. 4.11: Проверка работоспособности скомпонованной программы

Убедившись в корректности работы программы, копирую рабочие файлы в свой локальный репозиторий. (рис. 4.12)

```
avtrofimov@tva-al:~/work/arh-pc/lab04$ cp hello.asm lab4.asm ../../study/2025-2026/"Архитектура компью
study_2025-2026_arh-pc/labs/lab04/
avtrofimov@tva-al:~/work/arh-pc/lab04$ cd ../../study/2025-2026/"Архитектура компьютера"/study_2025-20
-pc/labs/lab04/
avtrofimov@tva-al:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$ ls
hello.asm lab4.asm
```

Рис. 4.12: Отправка файлов в локальный репозиторий

Загрузка изменений на свой удаленный репозиторий на GitHub. (рис. 4.13)

```
avtrofimov@tva-al:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$
git add .
avtrofimov@tva-al:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$
git status
На ветке master
Ваша ветка обновлена в соответствии с «origin/master».
Изменения, которые будут включены в коммит:
  (используйте «git reset HEAD <файл>...», чтобы убрать из индекса)

    новый файл:   hello.asm
    новый файл:   lab4.asm

avtrofimov@tva-al:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$
git commit -m "feat(main): upload 4 lab work"
[master 86b1712] feat(main): upload 4 lab work
 2 files changed, 36 insertions(+)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab4.asm

avtrofimov@tva-al:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$
git push
Подсчет объектов: 7, готово.
Delta compression using up to 16 threads.
Сжатие объектов: 100% (7/7), готово.
Запись объектов: 100% (7/7), 808 bytes | 0 bytes/s, готово.
Total 7 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To github.com:tremdim07/study_2025-2026_arh-pc.git
   a8ed864..a640251 master -> master
```

Рис. 4.13: Загрузка изменений

5 Выводы

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

6 Список литературы

1. [Курс на ТУИС](#)
2. [4. Лабораторная работа №4. Создание и процесс обработки программ на языке ассемблера NASM](#)
3. [Программирование на языке ассемблера NASM Столяров А. В.](#)