

Отчёт по лабораторной работе №4

Дисциплина: архитектура компьютера

Черкашина Ангелина Максимовна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
4.1	Создание программы Hello world!	10
4.2	Работа с транслятором NASM	12
4.3	Работа с расширенным синтаксисом командной строки NASM . .	12
4.4	Работа с компоновщиком LD	13
4.5	Запуск исполняемого файла	13
4.6	Выполнение заданий для самостоятельной работы	14
5	Выводы	17
6	Список литературы	18

Список иллюстраций

4.1	Создание новой директории	10
4.2	Перемещение между директориями	10
4.3	Создание пустого файла	10
4.4	Открытие файла в текстовом редакторе	11
4.5	Заполнение файла	11
4.6	Компиляция текста программы	12
4.7	Компиляция текста программы	12
4.8	Передача объектного файла hello.o на обработку компоновщику .	13
4.9	Передача объектного файла obj.o на обработку компоновщику . .	13
4.10	Запуск исполняемого файла hello	13
4.11	Создание копии файла	14
4.12	Изменение программы	14
4.13	Компиляция текста программы	15
4.14	Передача объектного файла lab4.o на обработку компоновщику .	15
4.15	Запуск исполняемого файла lab4	15
4.16	Создание копий файлов в другом каталоге	15
4.17	Загрузка файлов на GitHub	16

Список таблиц

1 Цель работы

Целью данной лабораторной работы является 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 ЭВМ являются центральный процессор, память и периферийные устройства. 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 плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора 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

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

4.1 Создание программы Hello world!

Создаю каталог для работы с программами на языке ассемблера NASM (рис. 4.1).

```
amcherkashina@dk1n22 ~ $ mkdir -p ~/work/arch-pc/lab04  
amcherkashina@dk1n22 ~ $
```

Рис. 4.1: Создание новой директории

Перехожу в созданный каталог (рис. 4.2).

```
amcherkashina@dk1n22 ~ $ cd ~/work/arch-pc/lab04  
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $
```

Рис. 4.2: Перемещение между директориями

Создаю в текущем каталоге пустой текстовый файл hello.asm с помощью команды touch (рис. 4.3).

```
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $ touch hello.asm  
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $ ls  
hello.asm  
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $
```

Рис. 4.3: Создание пустого файла

Открываю созданный файл с помощью текстового редактора gedit (рис. 4.4).

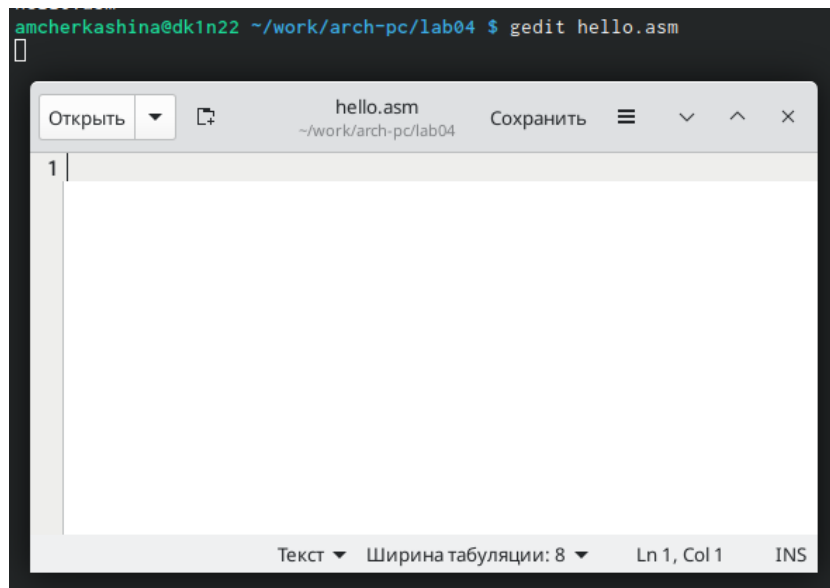


Рис. 4.4: Открытие файла в текстовом редакторе

Заполняю файл, вводя в него программу для вывода “Hello world!” (рис. 4.5).

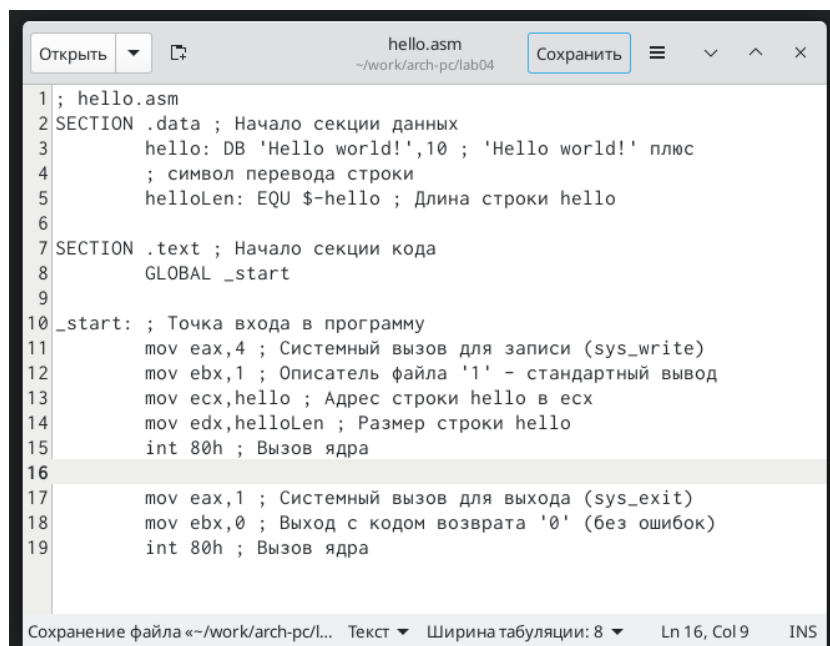


Рис. 4.5: Заполнение файла

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. Далее проверяю правильность выполнения команды с помощью `ls`. Действительно, создан файл “hello.o” (рис. 4.6).

```
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $
```

Рис. 4.6: Компиляция текста программы

4.3 Работа с расширенным синтаксисом командной строки NASM

Ввожу команду `nasm -o obj.o -f elf -g -l list.lst hello.asm`, которая скомпилирует файл `hello.asm` в файл `obj.o` (опция `-o` позволяет задать имя объектного файла, в данном случае `obj.o`), при этом формат выходного файла будет `elf`, и в него будут включены символы для отладки (ключ `-g`). Кроме того, будет создаваться файл листинга `list.lst` (опция `-l`). Снова проверяю корректность выполнения команды с помощью `ls` (рис. 4.7).

```
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $
```

Рис. 4.7: Компиляция текста программы

4.4 Работа с компоновщиком LD

Передаю объектный файл `hello.o` на обработку компоновщику LD, чтобы получить исполняемый файл `hello`. Использую ключ `-o`, чтобы задать имя создаваемого исполняемого файла. С помощью утилиты `ls` проверяю, что исполняемый файл `hello` был создан (рис. 4.8).

```
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $
```

Рис. 4.8: Передача объектного файла `hello.o` на обработку компоновщику

Выполняю следующую команду: `ld -m elf_i386 obj.o -o main` (рис. 4.9). 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`

```
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $
```

Рис. 4.9: Передача объектного файла `obj.o` на обработку компоновщику

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл `hello`, находящийся в текущем каталоге (рис. 4.10).

```
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $ ./hello
Hello world!
amcherkashina@dk1n22 ~/work/arch-pc/lab04 $
```

Рис. 4.10: Запуск исполняемого файла `hello`

4.6 Выполнение заданий для самостоятельной работы

1. С помощью команды `cp` в каталоге `~/work/arch-pc/lab04` создаю копию файла `hello.asm` с именем `lab4.asm` (рис. 4.11).

```
amcherkashina@dk3n40 ~ $ cd ~/work/arch-pc/lab04
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $
```

Рис. 4.11: Создание копии файла

2. С помощью текстового редактора `gedit` открываю файл `lab4.asm` и вношу изменения в текст программы так, чтобы вместо “Hello world!” она выводила на экран строку с моим именем и фамилией (рис. 4.12).

```
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $ gedit lab4.asm
Открыть  lab4.asm  Сохранить
~/work/arch-pc/lab04

1 ; lab4.asm
2 SECTION .data ; Начало секции данных
3     lab4: DB 'Angelina Cherkashina',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 вызов для выхода (sys_exit)
18     mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
19     int 80h ; Вызов ядра
```

Рис. 4.12: Изменение программы

3. Компилирую текст программы в объектный файл. С помощью команды `ls` проверяю, что файл `lab4.o` создан (рис. 4.13).

```
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $
```

Рис. 4.13: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. 4.14).

```
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $
```

Рис. 4.14: Передача объектного файла lab4.o на обработку компоновщику

Запускаю получившийся исполняемый файл lab4. На экран действительно выводятся мои имя и фамилия (рис. 4.15).

```
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $ ./lab4
Angelina Cherkashina
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $
```

Рис. 4.15: Запуск исполняемого файла lab4

4. Копируйте файлы hello.asm и lab4.asm в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/ с помощью команды cp. С помощью ls убеждаюсь в правильности выполнения команды (рис. 4.16).

```
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $ cp hello.asm ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/
amcherkashina@dk3n40 ~/work/arch-pc/lab04 $ cd ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/
amcherkashina@dk3n40 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello.asm lab4.asm presentation report
amcherkashina@dk3n40 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Рис. 4.16: Создание копий файлов в другом каталоге

С помощью команд `git add .` и `git commit` добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №4. Отправляю файлы на сервер с помощью команды `git push` (рис. 4.17).

```
amcherkashina@dk3n40 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git add .
amcherkashina@dk3n40 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git commit -m "Add files for lab04"
[master 9246d83] Add files for lab04
2 files changed, 38 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
amcherkashina@dk3n40 ~/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 4 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.05 KiB | 1.05 MiB/c, 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:angelinacherkashina/study_2023-2024_arc-pc.git
 e9c9b7b..9246d83  master -> master
amcherkashina@dk3n40 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Рис. 4.17: Загрузка файлов на GitHub

5 Выводы

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

6 Список литературы

1. Архитектура ЭВМ