

Отчёт по лабораторной работе №3

Дисциплина: архитектура компьютера

Репкина Елизавета Андреевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
5	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	создание и переход в каталог	10
4.2	создание текстового файла	10
4.3	начало редактирования файла	11
4.4	работа в gedit	11
4.5	создание объектного файла	12
4.6	компиляция исходного файла	12
4.7	передаю объектный файл на обработку компоновщику	12
4.8	запуск	13
4.9	копирование файла	13
4.10	изменение текста программы	13
4.11	запуск исполняемого файла	14
4.12	перемещение файлов	14
4.13	загрузка файлов на github	15

Список таблиц

1 Цель 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 электронной вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства. Все эти компоненты взаимодействуют друг с другом через общую шину, к которой они подключены. Физически шина представляет собой множество проводников, соединяющих различные устройства. В современных компьютерах эти проводники выполнены в виде электропроводящих дорожек на материнской плате.

Центральный процессор (ЦП) выполняет обработку информации и координирует работу всех узлов компьютера. Он состоит из следующих основных компонентов: - Арифметико-логическое устройство (АЛУ): выполняет логические и арифметические операции, необходимые для обработки данных, хранящихся в памяти. - Устройство управления (УУ): отвечает за управление всеми устройствами ЭВМ. - Регистры: это сверхбыстрая память небольшого объема, используемая для временного хранения промежуточных результатов выполнения инструкций. Регистры делят на два типа: общего назначения и специальные.

Для программирования на ассемблере важно знать, какие регистры существуют и как ими пользоваться. В большинстве команд, написанных на ассемблере, используются регистры в качестве операндов. Основные операции представляют собой преобразование данных, хранящихся в регистрах, такие как перенаправление данных между регистрами и памятью или выполнение арифметических и логических операций. Доступ к регистрам осуществляется по именам, а не по адресам, как в основной памяти. Каждый регистр процессора архитектуры x86

имеет имя, состоящее из двух или трех букв латинского алфавита. Примеры регистров общего назначения: - 64-битные: RAX, RCX, RDX, RBX, RSI, RDI - 32-битные: EAX, ECX, EDX, EBX, ESI, EDI - 16-битные: AX, CX, DX, BX, SI, DI - 8-битные: AH, AL, CH, CL, DH, DL, BH, BL

Другим важным компонентом ЭВМ является оперативное запоминающее устройство (ОЗУ). Это быстродействующее энергозависимое хранилище, которое взаимодействует с центральным процессором и используется для хранения программ и данных, находящихся в активной работе. ОЗУ состоит из одинаковых пронумерованных ячеек памяти, каждая из которых имеет свой адрес.

Периферийные устройства ЭВМ можно разделить на следующие группы: - Устройства внешней памяти: предназначены для долговременного хранения больших объемов данных. - Устройства ввода-вывода: обеспечивают взаимодействие ЦП с внешней средой.

В основе работы ЭВМ лежит принцип программного управления, что означает, что компьютер решает задачу, выполняя последовательность действий, записанных в программе. Код машинной команды состоит из множества двоичных значений (0 и 1) и можно выделить две основные части: 1. Операционную часть, которая содержит код команды для выполнения. 2. Адресную часть, где хранятся данные или адреса, участвующие в операции.

Во время выполнения каждой команды процессор проходит через стандартный процесс, называемый командным циклом, который включает следующие шаги: 1. Формирование адреса следующей команды в памяти. 2. Считывание кода команды и ее дешифровка. 3. Выполнение команды. 4. Переход к следующей команде.

Язык ассемблера (assembly language, сокращенно asm) – это низкоуровневый язык, ориентированный на машинные команды. NASM (Netwide Assembler) является открытым проектом, предоставляющим различные версии ассемблера для разных операционных систем и позволяющим получать объектные файлы для этих систем. NASM использует синтаксис Intel и поддерживает инструкции

x86-64.

4 Выполнение лабораторной работы

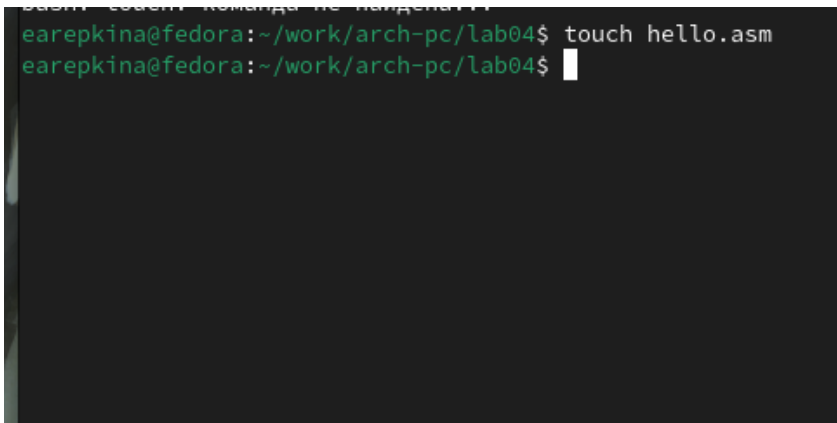
Программа Hello world! Создаю каталог для работы с программами на языке ассемблера NASM и перехожу в созданный каталог (рис. 4.1)



```
earepkina@fedora:~$ mkdir -p ~/work/arch-pc/lab04
earepkina@fedora:~$ cd ~/work/arch-pc/lab04
earepkina@fedora:~/work/arch-pc/lab04$
```

Рис. 4.1: создание и переход в каталог

Создаю текстовый файл с именем hello.asm (рис. 4.2)



```
ваша команда не найдена
earepkina@fedora:~/work/arch-pc/lab04$ touch hello.asm
earepkina@fedora:~/work/arch-pc/lab04$
```

Рис. 4.2: создание текстового файла

открываю этот файл с помощью любого текстового редактора, например, gedit (рис. 4.3)

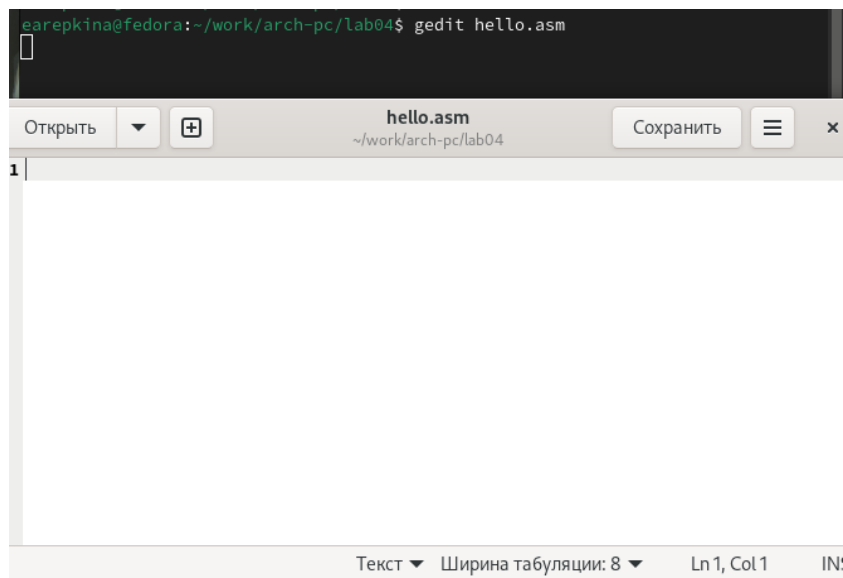


Рис. 4.3: начало редактирования файла

и ввожу в него текст:(рис. 4.4)

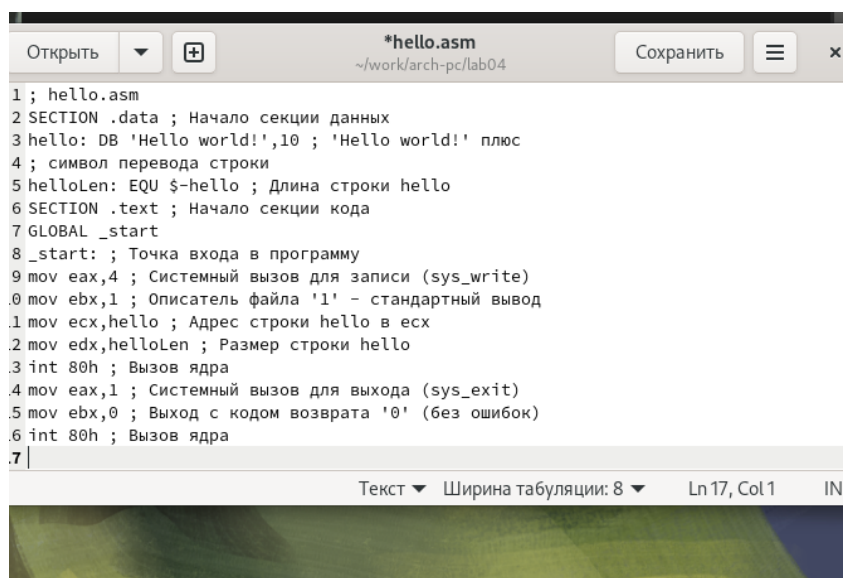


Рис. 4.4: работа в gedit

Транслятор NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World»

необходимо написать `nasm -f elf hello.asm` и с помощью команды `ls` проверьте, что объектный файл был создан (рис. 4.5)

```
earepkina@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
earepkina@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
```

Рис. 4.5: создание объектного файла

Расширенный синтаксис командной строки NASM

Выполняю указанную в задании команду (рис. 4.6)

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

Рис. 4.6: компиляция исходного файла

Компоновщик LD Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику (рис. 4.7)

```
earepkina@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
earepkina@fedora:~/work/arch-pc/lab04$ ды
bash: ды: команда не найдена...
earepkina@fedora:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.7: передаю объектный файл на обработку компоновщику

Компоновщик `ld` не предполагает по умолчанию расширений для файлов, но принято использовать следующие расширения: • `.o` – для объектных файлов; • без расширения – для исполняемых файлов; • `.tar` – для файлов схемы программы; • `.lib` – для библиотек. Ключ `-o` с последующим значением задаёт в данном случае имя создаваемого исполняемого файла. Выполните указанную в задании команду (рис. ??)

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

Ответ на вопросы: Ис-

полняемый 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`

Запуск исполняемого файла(рис. 4.8)

```
earepkina@fedora:~/work/arch-pc/lab04$ ./hello
Hello world!
earepkina@fedora:~/work/arch-pc/lab04$
```

Рис. 4.8: запуск

Задание для самостоятельной работы 1. В каталоге ~/work/arch-pc/lab04 с помощью команды `cp` создайте копию файла `hello.asm` с именем `lab4.asm` (рис. 4.9)

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

Рис. 4.9: копирование файла

2. С помощью любого текстового редактора внесите изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с вашими фамилией и именем. (рис. 4.10)

```
; hello.asm
SECTION .data ; Начало секции данных
hello: DB 'Репкина Елизавета!',10 ; 'Hello world!' плюс
; символ перевода строки
helloLen: EQU $-hello ; Длина строки hello
SECTION .text ; Начало секции кода
GLOBAL _start
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,hello ; Адрес строки hello в ecx
mov edx,helloLen ; Размер строки hello
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
int 80h ; Вызов ядра
```

Рис. 4.10: изменение текста программы

3. Оттранслируйте полученный текст программы `lab4.asm` в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл. (рис. 4.11)

```
hello.o hello.asm lab4.o lab4.asm list.lst main obj.o
earepkina@fedora:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
earepkina@fedora:~/work/arch-pc/lab04$ ls
hello.o lab4.o lab4.asm list.lst main obj.o
earepkina@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
earepkina@fedora:~/work/arch-pc/lab04$ ls
hello.o lab4.o lab4.asm list.lst main obj.o
earepkina@fedora:~/work/arch-pc/lab04$ ./lab4
Репкина Елизавета!
earepkina@fedora:~/work/arch-pc/lab04$
```

Рис. 4.11: запуск исполняемого файла

4. Скопируйте файлы `hello.asm` и `lab4.asm` в Ваш локальный репозиторий в каталог `~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/`. (рис. 4.12)

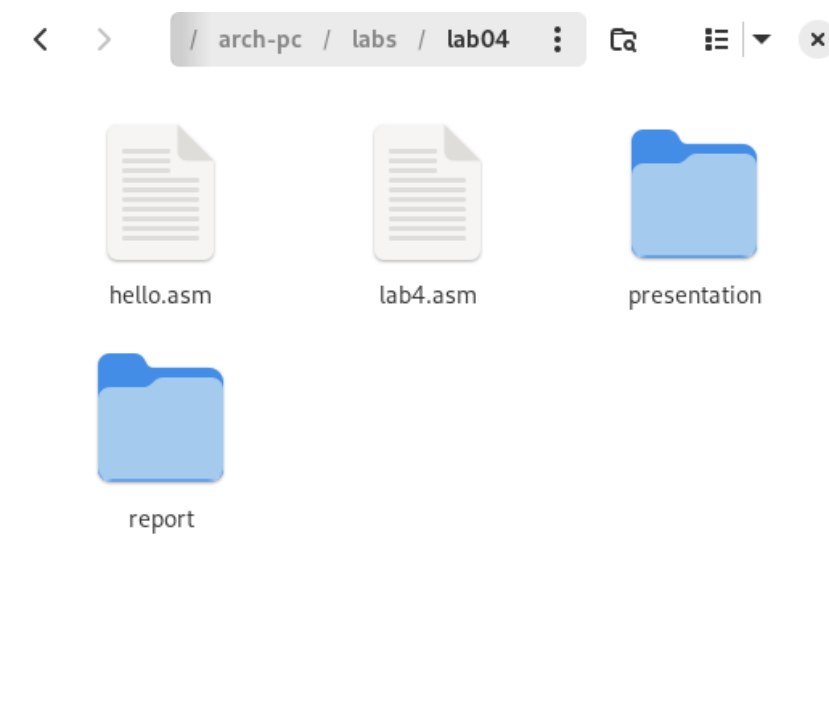


Рис. 4.12: перемещение файлов

Загрузите файлы на Github. (рис. 4.13)

```

# $ git add .
earepkina@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git commit -am 'feat(main): add files lab-4'
[master 2fala3f] feat(main): add files lab-4
5 files changed, 219 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
create mode 100644 labs/lab04/report/image/1.png
create mode 100644 labs/lab04/report/image/2.png
create mode 100644 labs/lab04/report/Л04_Репкина_отчет.md
earepkina@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git push
Перечисление объектов: 16, готово.
Подсчет объектов: 100% (16/16), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (11/11), готово.
Запись объектов: 100% (11/11), 29.99 КиБ | 6.00 МБ/с, готово.
Total 11 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.
To github.com:lizeew/study_2024-2025_arch-pc.git
 853a004..2fala3f master -> master

```

Рис. 4.13: загрузка файлов на github

5 Выводы

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

Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-

- е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
 17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
 18. — 1120 с. — (Классика Computer Science).