

Отчёт по лабораторной работе №4

дисциплина: Архитектура компьютера

Веретенников Дмитрий Олегович

Содержание

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	12
4.4	Компоновщик LD	12
4.5	Запуск исполняемого файла	13
4.6	Задания для самостоятельной работы	13
5	Выводы	15
	Список литературы	16

Список иллюстраций

4.1	Создание файла в директории и его открытие	10
4.2	Ввод кода	11
4.3	Компиляция программы	11
4.4	Проверка созданных файлов	12
4.5	Компиляция файла	12
4.6	Проверка созданных файлов	12
4.7	Отправка файла компановщику и проверка созданного файла . . .	12
4.8	Выполнение команды	13
4.9	Запуск исполняемого файла	13
4.10	Выполнение задания для самостоятельной работы	14

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 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

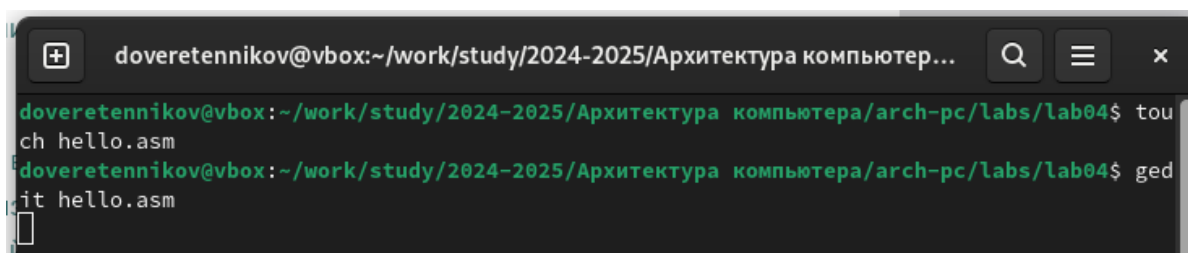
следующей команде.

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 Программа Hello world!

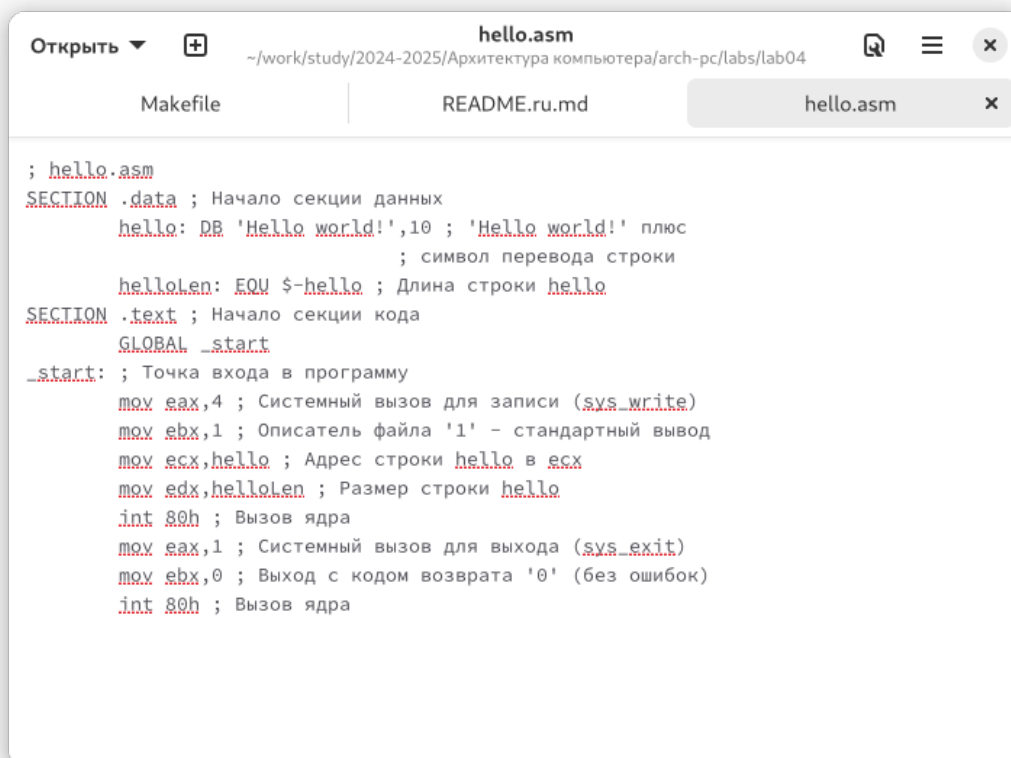
Захожу в созданный каталог lab04, создаю файл с именем hello.asm и открываю его с помощью gedit. (рис. 4.1).

A screenshot of a terminal window. The title bar shows the user 'doveretennikov@vbox' and the current directory '~/work/study/2024-2025/Архитектура компьютер...'. The terminal shows the following commands and their outputs:

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ touch hello.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ gedit hello.asm
```

Рис. 4.1: Создание файла в директории и его открытие

С помощью gedit ввожу код в файл. (рис. 4.2)



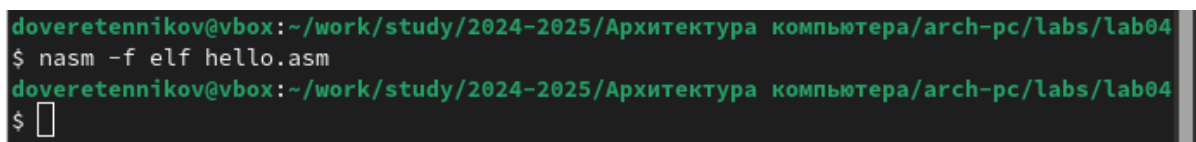
The screenshot shows a code editor window titled 'hello.asm'. The address bar indicates the file path: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04. The editor has tabs for 'Makefile', 'README.ru.md', and 'hello.asm'. The code is as follows:

```
; hello.asm
SECTION .data ; Начало секции данных
    hello: DB 'Hello world!',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. 4.2: Ввод кода

4.2 Транслятор NASM

Компилирую программу с помощью NASM (рис. 4.3) и проверяю с помощью команды ls, что файлы созданы. (рис. 4.4).



The screenshot shows a terminal window with the following commands and output:

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ nasm -f elf hello.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$
```

Рис. 4.3: Компиляция программы

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ ls
hello.asm hello.o report
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$
```

Рис. 4.4: Проверка созданных файлов

4.3 Расширенный синтаксис командной строки NASM

Компилирую файл (рис. 4.5) и проверяю с помощью команды ls, что все файлы созданы. (рис. 4.6).

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ nasm -o obj.o -f elf -g -l list.lst hello.asm
```

Рис. 4.5: Компиляция файла

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ ls
hello.asm hello.o list.lst obj.o report
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$
```

Рис. 4.6: Проверка созданных файлов

4.4 Компоновщик LD

С помощью команды ld передаю объектный файл на обработку компоновщику и с помощью команды ls убеждаюсь что файл был создан. (рис. 4.7)

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ ld -m elf_i386 hello.o -o hello
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ ls
hello hello.asm hello.o list.lst obj.o report
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$
```

Рис. 4.7: Отправка файла компоновщику и проверка созданного файла

Выполняю следующую команду, указанную в лабораторной работе, результатом исполнения команды будет созданный файл `main`, скомпонованный из объектного файла `obj.o`. (рис. 4.8)

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ ld -m elf_i386 obj.o -o main
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o  report
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$
```

Рис. 4.8: Выполнение команды

4.5 Запуск исполняемого файла

Запускаю исполняемый файл из текущего каталога. (рис. 4.9)

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ ./hello
Hello world!
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$
```

Рис. 4.9: Запуск исполняемого файла

4.6 Задания для самостоятельной работы

Создаю копию файла `hello.asm` с именем `lab4.asm`, с помощью `gedit` вношу изменения в текст программы, далее транслирую текст программы `lab4.asm` в объектный файл, выполняю компоновку и запускаю полученный файл. (рис. 4.10)

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ cp hello.asm lab4.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o  report
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ gedit lab4.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ ./lab4.asm
bash: ./lab4.asm: Отказано в доступе
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ nasm -f elf lab4.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ ld -m elf_i386 lab4.o -o lab4
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$ ./lab4
Veretennikov Dmitriy!
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
$
```

Рис. 4.10: Выполнение задания для самостоятельной работы

Далее копирую файлы в локальный репозиторий и загружаю файлы на Github.

5 Выводы

После выполнения данной лабораторной работы освоил 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/Learning-bash-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. — БХВ- Петербург, 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).