

Лабораторная работа №4

Создание и процесс обработки программ на языке ассемблер NASM

Павленко Сергей

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	8
4	Выводы	11
5	Выполнение лабораторной работы	12
6	Выводы по самостоятельной работы	14
	Список литературы	15

Список иллюстраций

3.1	Рис.1	8
3.2	Рис.2	8
3.3	Рис.3	9
3.4	Рис.4	9
3.5	Рис.5	9
3.6	Рис.6	10
3.7	Рис.7	10
3.8	Рис.8	10
5.1	Рис.9	12
5.2	Рис.10	12
5.3	Рис.11	13
5.4	Рис.12	13
5.5	Рис.13	13
5.6	Рис.14	13

Список таблиц

1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM

2 Теоретическое введение

Основные принципы работы компьютера Основными функциональными элементами электронно-вычислительной машины 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, соединяющий 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: * List item 1 арифметико-логическое устройство (АЛУ) - выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти * List item 2 устройство управления (УУ) - обеспечивает управление и контроль всех устройств компьютера * List item 3 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 присутствуют. Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. В качестве примера приведем названия основных регистров общего назначения:

* List item 1 RAX, RCX, RDX, RBX, RSI, RDI - 64 битные

- * List item 2 EAX, ECX, EDX, EBX, ESI, EDI - 32 битные
- * List item 3 AX, CX, DX, BX, SI, DI - 16 битные
- * List item 4 AH, AL, CH, CL, DH, DL, BH, BL - 8 битные

Таким образом можно отметить, что вы можете написать в своей программе, например,

```
mov ax, 1 mov eax, 1
```

Обе команды поместят в регистр AX число 1. Разница будет заключаться только в том, то число, но не 1. А вот в регистре AX будет число 1.

Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ –

3 Выполнение лабораторной работы

Перейдём в каталог

`cd ~/work/arch-pc/lab04` Создадим текстовый файл с именем `hello.asm` `touch hello.asm` Откроем этот файл с помощью любого текстового редактора, например, `gedit` `gedit hello.asm`

```
spavlenko@spavlenko:~$ cd ~/work/study/2023-2024/Архитектура\ компьютера/study_2
023-2024_arhpc/
config/      .git/          labs/          presentation/ template/
spavlenko@spavlenko:~$ cd ~/work/study/2023-2024/Архитектура\ компьютера/study_2
023-2024_arhpc/labs/lab04/
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-202
4_arhpc/labs/lab04$ touch hello.asm
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-202
4_arhpc/labs/lab04$ gedit hello.asm
```

Рис. 3.1: Рис.1

Введём в него следующий текст



```
Open  v  ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/L... Save  E
hello.asm
1 SECTION .data
2     hello: DB 'Hello world!' ,10
3
4     helloLen: EQU $-hello
5
6 SECTION .text
7     Global _start
8
9     _start:
10
11         mov eax,4
12         mov ebx,1
13         mov ecx,hello
14         mov edx,helloLen
15
16         mov eax,1
17         mov ebx,0
18         int 80h
```

Рис. 3.2: Рис.2

NASM превращает текст программы в объектный код. Например, для компиляции приведё

`nasm -f elf hello.asm` Если текст программы набран без ошибок, то транслятор преобразует текст программы из файла `hello.asm` в объектный код, который запишется в файл `hello.o`. С помощью команды `ls` проверим, что объектный файл был создан.

```
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ nasm -f elf hello.asm
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ls
hello.asm hello.o presentation report
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 3.3: Рис.3

Выполним следующую команду:

`nasm -o obj.o -f elf -g -l list.lst hello.asm` С помощью команды `ls` проверим, что файлы были созданы

```
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ls
hello.asm hello.o list.lst obj.o presentation report
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 3.4: Рис.4

Как видно из схемы на пред. рис., чтобы получить исполняемую программу, объектный

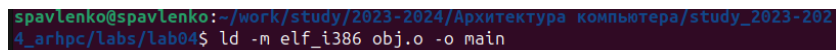
необходимо передать на обработку компоновщику: `ld -m elf_i386 hello.o -o hello` С помощью команды `ls` проверим, что файлы были созданы

```
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ld -m elf_i386 hello.o -o hello
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ls
hello hello.asm hello.o list.lst obj.o presentation report
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 3.5: Рис.5

Ключ `-o` с последующим значением задаёт в данном случае имя создаваемого исполняемого файла. Выполните следующую команду

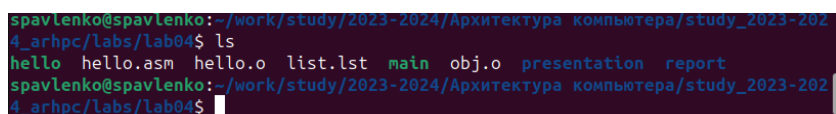
```
ld -m elf_i386 obj.o -o main
```



```
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ld -m elf_i386 obj.o -o main
```

Рис. 3.6: Рис.6

С помощью команды `ls` проверим, какое имя имеет объектный файл

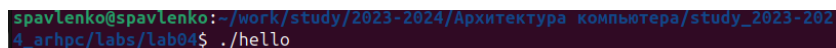


```
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ls
hello hello.asm hello.o list.lst main obj.o presentation report
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 3.7: Рис.7

Запустим на выполнение созданный исполняемый файл, находящийся в текущем каталоге

```
./hello
```



```
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ./hello
```

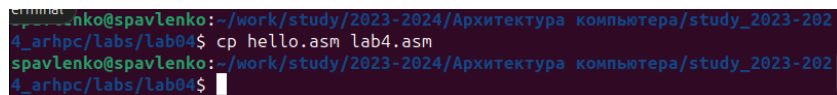
Рис. 3.8: Рис.8

4 Выводы

Выполнив данную лабораторную работу мы поняли, как можно пользоваться простейшими

5 Выполнение лабораторной работы

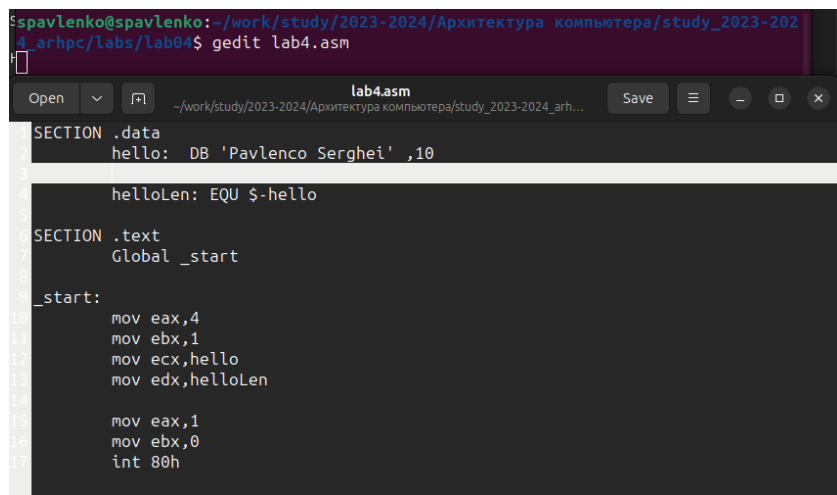
1. В каталоге ~/work/arch-pc/lab04 с помощью команды `cp` создадим копию файла `hello.asm` с именем `lab4.asm`



```
terminal
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ cp hello.asm lab4.asm
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 5.1: Рис.9

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.



```
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ gedit lab4.asm
lab4.asm
~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh... Save
1 SECTION .data
2     hello: DB 'Pavlenko Serghei' ,10
3
4     helloLen: EQU $-hello
5
6 SECTION .text
7     Global _start
8
9 _start:
10     mov eax,4
11     mov ebx,1
12     mov ecx,hello
13     mov edx,helloLen
14
15     mov eax,1
16     mov ebx,0
17     int 80h
```

Рис. 5.2: Рис.10

3. Оттранслируйте полученный текст программы `lab4.asm` в объектный файл.

```
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ nasm -o lab4.o -f elf -g -l lablist.lst lab4.asm
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ls
hello.o    lab4.o      list.lst   obj.o      report
hello.asm  lab4.asm    lablist.lst main        presentation
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 5.3: Рис.11

Выполним компоновку объектного файла

```
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ld -m elf_i386 lab4.o -o lab4
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ls
hello.o    lab4.asm    lablist.lst main        presentation
hello.asm  lab4        lab4.o      list.lst    obj.o      report
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 5.4: Рис.12

Запустите получившийся исполняемый файл.

```
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ./lab4
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 5.5: Рис.13

4. Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-rc/labs/lab04/. Загрузите файлы на Github

```
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04/report$ make
pandoc "report.md" --filter pandoc/filters/pandoc_fignos.py --filter pandoc/filters/pandoc_eqnos.py --filter pandoc/filters/pandoc_tablenos.py --filter pandoc/filters/pandoc_secnos.py --number-sections --citeproc -o "report.docx"
pandoc "report.md" --filter pandoc/filters/pandoc_fignos.py --filter pandoc/filters/pandoc_eqnos.py --filter pandoc/filters/pandoc_tablenos.py --filter pandoc/filters/pandoc_secnos.py --pdf-engine=lualatex --pdf-engine-opt=--shell-escape --citeproc --number-sections -o "report.pdf"
spavlenko@spavlenko:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04/report$
```

Рис. 5.6: Рис.14

6 Выводы по самостоятельной работы

С помощью данных заданий, мы на практике закрепили пройденный материал: по преобразовыванию файлов, компилированию кода, по передаче файла на обработку и запустили исходный файл.

Список литературы

1. GDB: The GNU Project Debugger. — URL: [https:// www.gnu.org/software/gdb/](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, 2015. — 1120 с. — (Классика Computer Science)