

Отчёт по лабораторной работе №4”

Дисциплина: Архитектура компьютера

Челухаев Кирилл Александрович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
4.1	Задания для самостоятельной работы	11
	Список литературы	14

Список иллюстраций

4.1	Создание файла hello.asm	10
4.2	Программа для вывода Hello world!	10
4.3	Исполняемый файл main	11
4.4	Запуск программы	11
4.5	Копия файла hello.asm	12
4.6	Измененная программа	12
4.7	Вывод измененной программы	13

Список таблиц

1 Цель работы

Цель данной лабораторной работы - освоить 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:

формирование адреса в памяти очередной команды; считывание кода коман-

ды из памяти и её дешифрация; выполнение команды; 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.

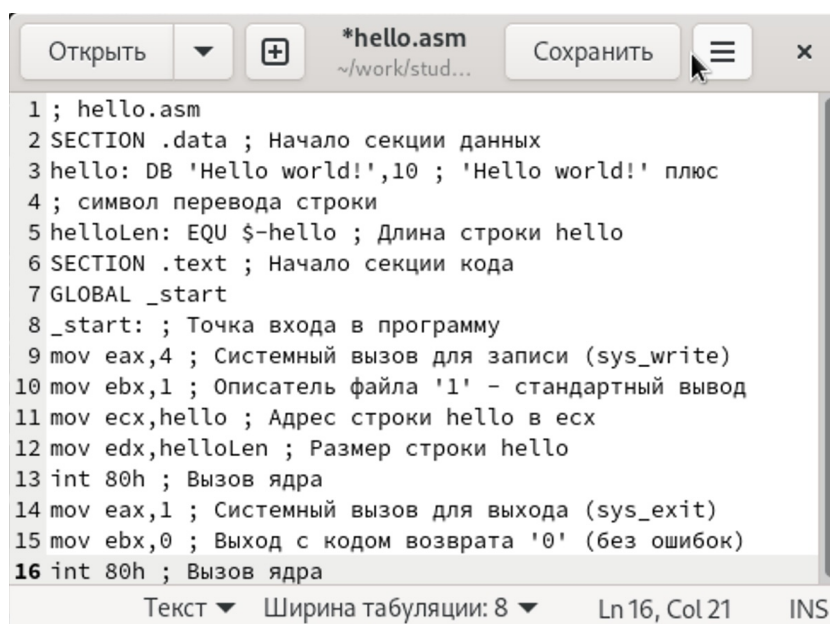
4 Выполнение лабораторной работы

Я создал каталог lab04 для работы с языком ассемблера NASM, и создал в нем файл hello.asm (рис. 4.1).

```
[KirillChel@fedora-linux-38 lab04]$ touch hello.asm
[KirillChel@fedora-linux-38 lab04]$ ls
hello.asm  presentation  report
[KirillChel@fedora-linux-38 lab04]$
```

Рис. 4.1: Создание файла hello.asm

я открыл файл и вставил в него программу для вывода Hello world! (рис. 4.2).



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

Рис. 4.2: Программа для вывода Hello world!

Далее я преобразую текст из файла в объектный код командой `nasm -f elf hello.asm`

Ввожу команду, которая скомпилирует файл hello.asm в файл obj.o, также с помощью ключа -l будет создан файл листинга list.lst

Далее я с помощью компоновщика ld я создаю исполняемый файл hello (рис.

```
??) /labs/lab04 $ nasm -f elf hello.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ ld -m elf_i386 hello.o -o hello
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ ls
```

Далее я выполнил команду ld -m elf_i386 obj.o -o main 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 (рис. 4.3)

```
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ ld -m elf_i386 obj.o -o main
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ ls
hello hello.asm hello.o list.lst main obj.o presentation report
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $
```

Рис. 4.3: Исполняемый файл main

Теперь я запустил на выполнение файл командой ./hello (рис. 4.4)

```
/labs/lab04 $ ./hello
Hello world!
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $
```

Рис. 4.4: Запуск программы

4.1 Задания для самостоятельной работы

Я создал копию файла hello.asm с названием lab04.asm (рис. 4.5)

```

kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ cp hello.asm lab04.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ ls
hello      hello.o    list.lst  obj.o      report
hello.asm  lab04.asm  main      presentation
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $

```

Рис. 4.5: Копия файла hello.asm

Далее я изменил программу так, чтобы она выводила мои имя и фамилию (рис. 4.6)

```

; lab04.asm
SECTION .data ; Начало секции данных
    lab04: DB 'Kirill Cheluhaev',10
    lab04Len: EQU $-lab04 ; Длина строки lab04
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,lab04; Адрес строки lab04 в ecx
    mov edx,lab04Len ; Размер строки lab04
    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. 4.6: Измененная программа

Компилирую текст файла в объектный файл и запускаю (рис. 4.7)

```

/labs/lab04 $ cp hello.asm lab04.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ ls
hello      hello.o    list.lst   obj.o      report
hello.asm  lab04.asm  main       presentation
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ nasm -f elf lab04.asm
lab04.asm:3: warning: unterminated string (missing `') [-w+pp-open-string]
]
lab04.asm:3: error: expression syntax error
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ nasm -f elf lab04.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ ld -m elf_i386 lab04.o -o lab04
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ ./lab04
Kirill Cheluhaev
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc
/labs/lab04 $ █

```

Рис. 4.7: Вывод измененной программы

Я загружаю все файлы на GitHub (рис. ??)

```

kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $
git add .
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $
git commit -m "Add lab04"
[master 7e45958] Add lab04
9 files changed, 48 insertions(+)
create mode 100755 labs/lab04/hello
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/hello.o
create mode 100755 labs/lab04/lab04
create mode 100644 labs/lab04/lab04.asm
create mode 100644 labs/lab04/lab04.o
create mode 100644 labs/lab04/list.lst
create mode 100755 labs/lab04/main
create mode 100644 labs/lab04/obj.o
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $
git push
Перечисление объектов: 16, готово.
Подсчет объектов: 100% (16/16), готово.
При сжатии изменений используется до 6 потоков
Загрузка объектов: 100% (13/13), готово.
Запись объектов: 100% (13/13), 3.36 КиБ | 3.36 МБ/с, готово.
total 13 (delta 7), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (7/7), completed with 2 local objects.
to github.com:Frog578nin/study_2024-2025_arch-pc.git
b5f991b..7e45958 master -> master
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $
█

```

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).