

Отчёт по лабораторной работе №4

Дисциплина: архитектура компьютера

Ермакова Анастасия Алексеевна

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	11
4.4	Компоновщик LD	11
4.5	Запуск исполняемого файла	12
4.6	Выполнение заданий для самостоятельной работы.	12
5	Выводы	15
6	Список литературы	16

Список иллюстраций

4.1	Создание пустого текстового файла.	10
4.2	Заполнение текстового файла.	10
4.3	Преобразование текстового файла в объектный код.	11
4.4	Компиляция файла.	11
4.5	Передача файла компоновщику LD на обработку.	12
4.6	Передача файла компоновщику LD на обработку.	12
4.7	Запуск исполняемого файла.	12
4.8	Создание копии файла.	12
4.9	Корректирование программы.	13
4.10	Компиляция файла.	13
4.11	Передача файла компоновщику LD на обработку.	13
4.12	Запуск исполняемого файла.	13
4.13	Отправка файлов на github.	14

Список таблиц

1 Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

1. Программа Hello world!
2. Транслятор NASM
3. Расширенный синтаксис командной строки NASM
4. Компоновщик 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 преобразова-

ние 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

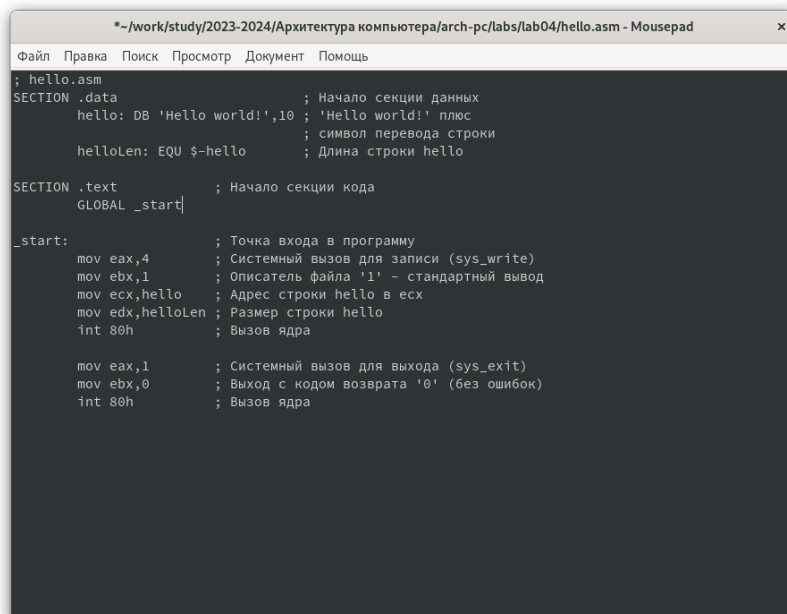
4.1 Программа Hello world!

С помощью утилиты `cd` перехожу в каталог, в котором буду выполнять работу. Затем с помощью утилиты `touch` создаю текстовый файл с именем `hello.asm`. Открываю его с помощью текстового редактора `mousepad`. (рис. 4.1)

```
anastasia@fedora: ~$ cd work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ touch hello.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ mousepad hello.asm
```

Рис. 4.1: Создание пустого текстового файла.

Ввожу в текстовый файл программу для вывода “Hello world!”. (рис. 4.2)



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/hello.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

; 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:
    mov eax,4                                ; Точка входа в программу
    mov ebx,1                                ; Системный вызов для записи (sys_write)
    mov ecx,hello                            ; Описатель файла '1' - стандартный вывод
    mov edx,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: Заполнение текстового файла.

4.2 Транслятор NASM

С помощью транслятора NASM превращаю текст программы в объектный код, для этого устанавливаю пакет `nasm` и далее прописываю команду `nasm -f elf hello.asm`. Проверяю, преобразовался ли файл. Должен появиться файл `hello.o`, проверяю это с помощью команды `ls`. (рис. 4.3)

```
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ nasm -f elf hello.asm
bash: nasm: команда не найдена...
Установить пакет «nasm», предоставляющий команду «nasm»? [N/y] y

* Ожидание в очереди...
* Загрузка списка пакетов....
Следующие пакеты должны быть установлены:
nasm-2.16.01-7.fc40.x86_64  A portable x86 assembler which uses Intel-like syntax
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...

anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ nasm -f elf hello.asm
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm  hello.o  presentation  report
```

Рис. 4.3: Преобразование текстового файла в объектный код.

4.3 Расширенный синтаксис командной строки NASM

Ввожу команду, которая скомпилирует исходный файл `hello.asm` в объектный файл `obj.o`, ключи `-g` и `-l` позволят включить в него символы для отладки и создать файл листинга `list.lst`. С помощью команды `ls` проверяю, что файлы действительно были созданы. (рис. 4.4)

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

Рис. 4.4: Компиляция файла.

4.4 Компоновщик LD

Передаю файл `hello.o` на обработку компоновщику `LD`, чтобы получить исполняемый файл `hello`. Ключ `-o` позволяет присвоить файлу имя `hello`. С помощью

команды `ls` убеждаюсь в правильности выполнения команды. (рис. 4.5)

```
anastasia@fedora: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
anastasia@fedora: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello hello.asm hello.o list.lst obj.o presentation report
```

Рис. 4.5: Передача файла компоновщику LD на обработку.

Выполняю следующую команду (рис. 4.6). Исполняемый файл будет иметь имя `main`, указанное после ключа `-o`. Объектный файл, из которого собран этот исполняемый файл, имеет имя `obj.o`.

```
anastasia@fedora: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ld -m elf_i386 obj.o -o main
anastasia@fedora: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello hello.asm hello.o list.lst main obj.o presentation report
```

Рис. 4.6: Передача файла компоновщику LD на обработку.

4.5 Запуск исполняемого файла

Запускаю на выполнение исполняемый файл `hello`, набрав в командной строке `./hello`. (рис. 4.7)

```
anastasia@fedora: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ./hello
Hello world!
```

Рис. 4.7: Запуск исполняемого файла.

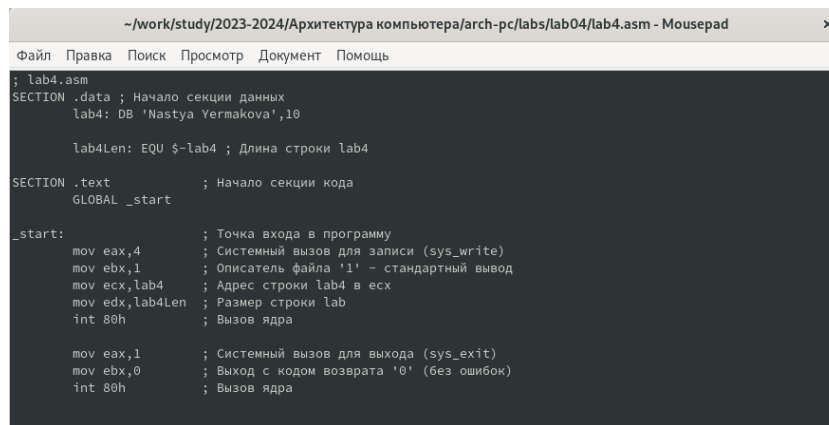
4.6 Выполнение заданий для самостоятельной работы.

С помощью команды `cp` создаю копию файла `hello.asm` с именем `lab4.asm`. Открываю файл с помощью текстового редактора `mousepad`. (рис. 4.8)

```
anastasia@fedora: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ cp hello.asm lab4.asm
anastasia@fedora: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ mousepad lab4.asm
```

Рис. 4.8: Создание копии файла.

Вношу изменения в текст программы в файле `lab4.asm`, чтобы на экран вывелась строка с моими именем и фамилией. (рис. 4.9)



```
~\work\study\2023-2024\Архитектура компьютера\arch-pc\labs\lab04\lab4.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

; lab4.asm
SECTION .data ; Начало секции данных
    lab4: DB 'Nastya Yermakova',10

    lab4Len: EQU $-lab4 ; Длина строки lab4

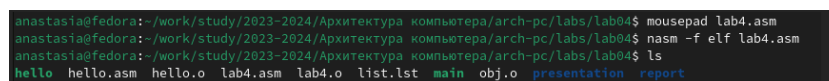
SECTION .text ; Начало секции кода
GLOBAL _start

_start:
    mov eax,4 ; Системный вызов для записи (sys_write)
    mov ebx,1 ; Описатель файла '1' - стандартный вывод
    mov ecx,lab4 ; Адрес строки lab4 в ecx
    mov edx,lab4Len ; Размер строки lab
    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. 4.9: Корректирование программы.

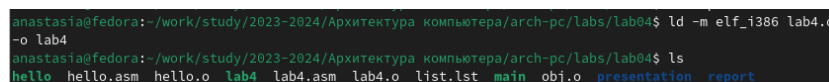
Компилирую файл программы в объектный файл, с помощью команды `ls` проверяю, создался ли новый файл `lab4.o`. (рис. 4.10)



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ mousepad lab4.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ nasm -f elf lab4.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o  presentation  report
```

Рис. 4.10: Компиляция файла.

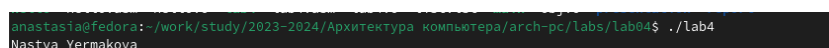
Передаю объектный файл `lab4.o` компоновщику LD на обработку, получаю файл `lab4`. Убеждаюсь в этом командой `ls`. (рис. 4.11)



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ld -m elf_i386 lab4.o
-o lab4
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o  presentation  report
```

Рис. 4.11: Передача файла компоновщику LD на обработку.

Запускаю получившийся исполняемый файл. (рис. 4.12)



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ./lab4
Nastya Yermakova
```

Рис. 4.12: Запуск исполняемого файла.

Наконец отправляю все файлы на GitHub. (рис. 4.13)

```

anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git add .
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git commit -m "Add files for lab04"
[master ba7998c] Add files for lab04
12 files changed, 107 insertions(+), 32 deletions(-)
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/lab4
create mode 100644 labs/lab04/lab4.asm
create mode 100644 labs/lab04/lab4.o
create mode 100644 labs/lab04/list.lst
create mode 100755 labs/lab04/main
create mode 100644 labs/lab04/obj.o
create mode 100644 labs/lab04/report/report.docx
create mode 100644 labs/lab04/report/report.pdf
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git push
Перечисление объектов: 22, готово.
Подсчет объектов: 100% (22/22), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (17/17), готово.
Запись объектов: 100% (17/17), 545.14 КиБ | 3.26 МБ/с, готово.
Total 17 (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:aannyaal/study_2023-2024_arh-pc.git
   3c1ld1f..ba7998c  master -> master
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$

```

Рис. 4.13: Отправка файлов на github.

5 Выводы

В ходе выполнения данной лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

6 Список литературы

1. Архитектура ЭВМ