

# Архитектура компьютера

## Отчёт по лабораторной работе №4

Лю Сяо

### 1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM

### 2 Задание

- 1) Создать программу Hello world
- 2) Работа с транслятором NASM
- 3) Работа с расширенным синтаксисом командой строки NASM
- 4) Работа с компоновщиком LD
- 5) Запуск исполняемого файла
- 6) Выполнение заданий для самостоятельной работы

### 3 Теоретическое введение

Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора (ЦП) 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. В процессе создания ассемблерной программы можно выделить четыре шага: • Набор текста программы в текстовом редакторе и сохранение её в отдельном файле. Каждый файл имеет свой тип (или расширение), который определяет назначение файла. Файлы с исходным текстом программ на языке ассемблера имеют тип `asm`. • Трансляция — преобразование с помощью транслятора, например `nasm`, текста программы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную дополнительную информацию, созданную транслятором. Тип объектного файла — `o`, файла листинга — `lst`. • Компоновка или линковка — этап обработки объектного кода

компоновщиком (ld), который принимает на вход объектные файлы и собирает по ним исполняемый файл. Исполняемый файл обычно не имеет расширения. Кроме того, можно получить файл карты загрузки программы в ОЗУ, имеющий расширение 7mar.

- Запуск программы. Конечной целью является работоспособный исполняемый файл. Ошибки на предыдущих этапах могут привести к некорректной работе программы, поэтому может присутствовать этап отладки программы при помощи специальной программы — отладчика. При нахождении ошибки необходимо провести коррекцию программы, начиная с первого шага. 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 (половинки 16-битных регистров).

Например, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX. В состав ЭВМ также входят периферийные устройства, которые можно разделить на:

- устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных (жёсткие диски, твердотельные накопители, магнитные ленты);
- 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. Программа состоит из машинных команд, которые указывают, какие операции и над какими данными (или операндами), в какой последовательности необходимо выполнить. Набор машинных команд определяется устройством конкретного процессора. 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. В самом общем виде 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.

## 4 Выполнение лабораторной работы

1) Создаю рекурсивно вложенные в папку work папки arch-pc и lab04, проверяю их создание

2) Перехожу в созданную папку

```
liveuser@localhost-live:/home$ cd ~/work/study/2023-2024/Архитектура\ компьютера  
/study_2023-2024_arhpc/labs/lab04/
```

Рис. 4.2: Переход в созданную папку

3) Создаю файл hello с разрешением asm и проверяю его создание

```
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ touch hello.asm  
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 4.3: Создание файла

4) Открываю этот файл в папо и копирую туда код из задания лабораторной работы

```
GNU nano 7.2 hello.asm  
; 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
```

[ Mark Set ]

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^N Replace	^U Paste	^J Justify	^_ Go To Line

Рис. 4.4: Заполнение файла

5) Скачиваю nasm

```
-2024_arhpc/labs/lab04$ sudo yum install nasm
Last metadata expiration check: 0:48:45 ago on Sat 26 Oct 2024 03:42:49 PM EDT.
Dependencies resolved.
=====
Package           Architecture      Version           Repository        Size
=====
Installing:
nasm               x86_64            2.16.01-7.fc40    fedora            356 k
=====
Transaction Summary
=====
Install 1 Package

Total download size: 356 k
Installed size: 2.5 M
Is this ok [y/N]: y
Downloading Packages:
nasm-2.16.01-7.fc40.x86_64.rpm                2.6 MB/s | 356 kB      00:00
-----
Total                                          412 kB/s | 356 kB      00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :                                1/1
  Installing     : nasm-2.16.01-7.fc40.x86_64    1/1
  Running scriptlet: nasm-2.16.01-7.fc40.x86_64    1/1

Installed:
nasm-2.16.01-7.fc40.x86_64
```

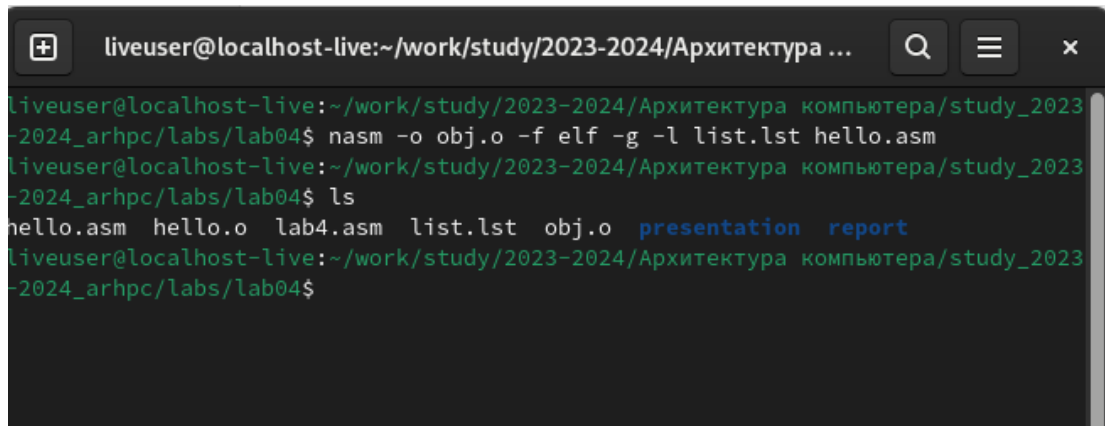
Рис. 4.5: Скачивание

6) Преобразовываю файл hello.asm в объектный код, записанный в файл hello.o. Проверяю, был ли создан файл

```
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023
-2024_arhpc/labs/lab04$ nasm -f elf hello.asm
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023
-2024_arhpc/labs/lab04$ ls
hello.asm hello.o lab4.asm presentation report
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023
-2024_arhpc/labs/lab04$
```

Рис. 4.6: Преобразование файла в объектный код

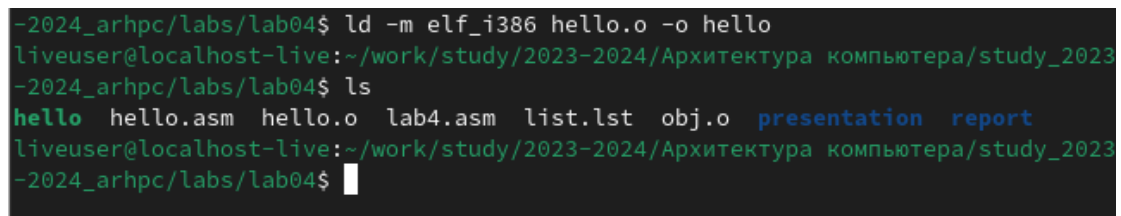
7) Преобразую файл hello.asm в obj.o с помощью опции -o, которая позволяет задать имя объекта. Из-за elf -g формат выходного файла будет elf, и в него будут включены символы для отладки, а так же будет создан файл листинга list.lst, благодаря -l. Проверяю созданные файлы



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

Рис. 4.7: Преобразование файла

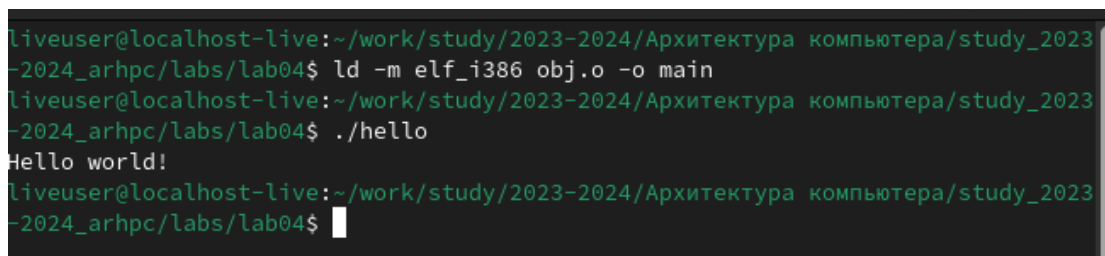
8) Передаю файл компоновщику с помощью ld. Проверяю, создан ли исполняемый файл



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

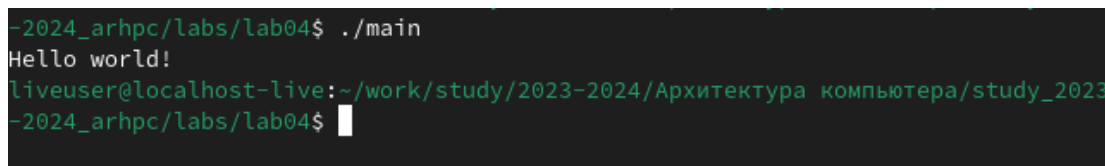
Рис. 4.8: Передача файла на обработку

9) Передаю компоновщику файл obj.o и называю скомпонованный файл main. запуская сначала код для предыдущего файла(1), а затем для созданного сейчас(2)



```
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ld -m elf_i386 obj.o -o main
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ./hello
Hello world!
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 4.9: 1



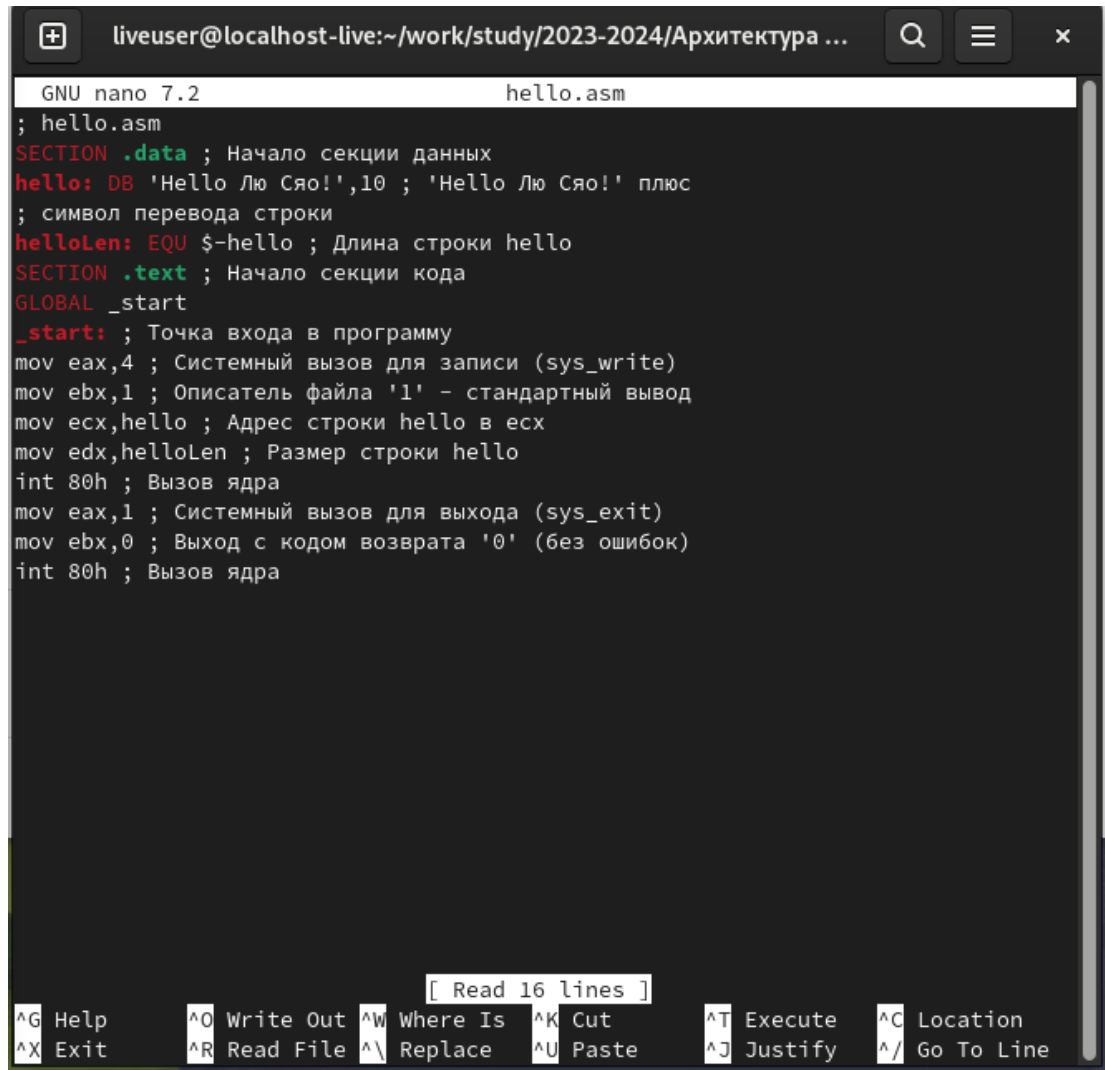
```
-2024_arhpc/labs/lab04$ ./main
Hello world!
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 4.9: 2

## 5 Выполнение заданий для самостоятельной работы

1) Копирую hello.asm с названием lab4.asm !Копирование файла](image/10.jpg){#ffg:001 width=70%}

2) С помощью nano изменяю текст кода так, чтобы он выводил моё имя и фамилию

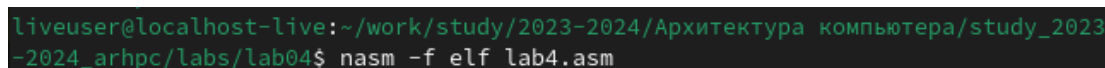


```
GNU nano 7.2 hello.asm
; hello.asm
SECTION .data ; Начало секции данных
hello: DB 'Hello Лю Сяо!',10 ; 'Hello Лю Сяо!' плюс
; символ перевода строки
helloLen: EQU $-hello ; Длина строки hello
SECTION .text ; Начало секции кода
GLOBAL _start
_start: ; Точка входа в программу
mov eax,4 ; Системный 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

[ Read 16 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

Рис. 5.1: Файл в nano

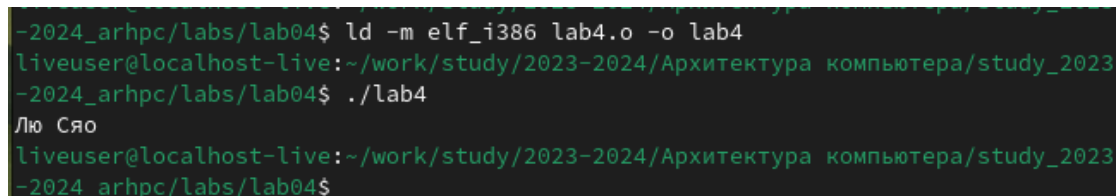
3) Транслирую файл lab4.asm в объектный



```
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ nasm -f elf lab4.asm
```

Рис. 5.2: Транслирую файл

4) Выполняю компоновку и запускаю исполняемый файл



```
-2024_arhpc/labs/lab04$ ld -m elf_i386 lab4.o -o lab4
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$ ./lab4
Лю Сяо
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/labs/lab04$
```

Рис. 5.3: Компоновка и исполнение

5) Выгружаю изменения на GitHub

```

-2024_arhpc$ git add .
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023
-2024_arhpc$ git commit -m "add file"
[master 7b64862] add file
 9 files changed, 19 insertions(+), 2 deletions(-)
 create mode 100755 labs/lab04/hello
 create mode 100644 labs/lab04/hello.o
 create mode 100755 labs/lab04/lab4
 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
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023
-2024_arhpc$ git pull
Already up to date.
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023
-2024_arhpc$ git push

```

Рис. 5.5: Выгружаю изменения

6) Копирую файл с отчётом и начинаю его заполнять

```

config/      .git/      labs/      presentation/ template/
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023
-2024_arhpc$ cd labs/lab04/report/
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023
-2024_arhpc/labs/lab04/report$ cp report.md 04.md
liveuser@localhost-live:~/work/study/2023-2024/Архитектура компьютера/study_2023
-2024_arhpc/labs/lab04/report$

```

Рис. 5.6: Копирование файла с отчётом

## 6 Выводы

Я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM