

# **Отчёт по лабораторной работа №4**

**Дисциплина: Компьютерные науки и технологии программирования**

ДЫМОВА Д.Д.

# Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выполнение заданий для самостоятельной работы	10
5	Выводы	12

# Список иллюстраций

3.1	Создание каталога . . . . .	7
3.2	Перемещение между директориями, создание файла и открытие с помощью текстового редактора . . . . .	7
3.3	Текстовый редактор с командой . . . . .	7
3.4	Компиляция текста программы . . . . .	8
3.5	Компиляция файла и проверка выполненной программы . . . . .	8
3.6	Передача файла на обработку компоновщику . . . . .	8
3.7	Передача файла на обработку компоновщику . . . . .	8
3.8	Запуск исполняемого файла . . . . .	8
3.9	Формат командной строки . . . . .	9
3.10	Запуск исполняемого файла . . . . .	9
4.1	Создание копии файла . . . . .	10
4.2	Трансляция и компоновка файлов . . . . .	11
4.3	Копирование файлов . . . . .	11
4.4	Загрузка данных . . . . .	11

## **Список таблиц**

# 1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

## 2 Теоретическое введение

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня, таких как C/C++, Perl, Python и пр.

Типичный формат записи команд NASM имеет вид: [метка:] мнемокод [операнд {, операнд}] [; комментарий]

Здесь мнемокод — непосредственно мнемоника инструкции процессору, которая является обязательной частью команды. Операндами могут быть числа, данные, адреса регистров или адреса оперативной памяти. Метка — это идентификатор, с которым ассемблер ассоциирует некоторое число, чаще всего адрес в памяти. Т.о. метка перед командой связана с адресом данной команды.

Допустимыми символами в метках являются буквы, цифры, а также следующие символы: `,`, `$`, `#`, `@`, `~`, `.` и `?` *Начинаться метка или идентификатор могут с буквы, `.`, и `?`.* Максимальная длина идентификатора 4095 символов.

Директивы — инструкции, не переводящиеся непосредственно в машинные команды, а управляющие работой транслятора.

## 3 Выполнение лабораторной работы

Я создаю каталог для работы с программами на языке ассемблер NASM (рис. 3.1).

```
dddihmova@dk5n60 ~ $ cd ~/work/arch-pc/lab04
```

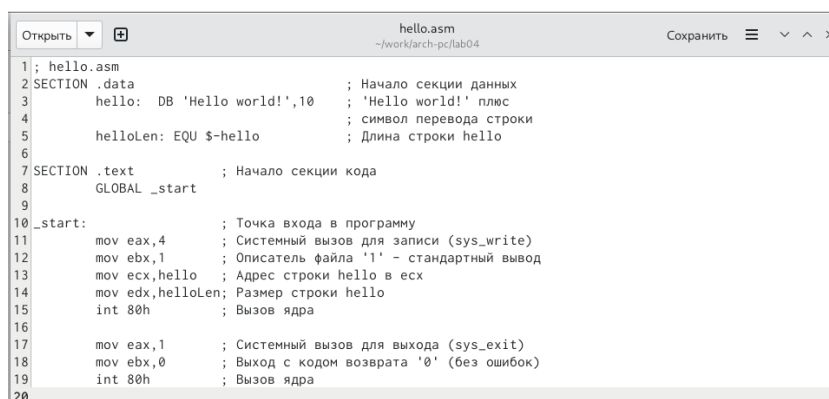
Рис. 3.1: Создание каталога

Перехожу в каталог и создаю текстовый файл с именем hello.asm. Открываю файл с помощью текстового редактора gedit (рис. 3.2).

```
dddihmova@dk5n60 ~ $ cd ~/work/arch-pc/lab04
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ touch hello.asm
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 3.2: Перемещение между директориями, создание файла и открытие с помощью текстового редактора

Ввожу программу в ассемблер и сохраняю (рис. 3.3).



```
1; hello.asm
2SECTION .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
7SECTION .text          ; Начало секции кода
8    GLOBAL _start
9
10 _start:
11        mov eax,4        ; Системный вызов для записи (sys_write)
12        mov ebx,1        ; Описатель файла '1' - стандартный вывод
13        mov ecx,hello     ; Адрес строки hello в ecx
14        mov edx,helloLen  ; Размер строки hello
15        int 80h          ; Вызов ядра
16
17        mov eax,1        ; Системный вызов для выхода (sys_exit)
18        mov ebx,0        ; Выход с кодом возврата '0' (без ошибок)
19        int 80h          ; Вызов ядра
20
```

Рис. 3.3: Текстовый редактор с командой

Компилирую текст программы “Hello world” и провожу проверку правильности созданного файла с помощью команды ls (рис. 3.4).

```
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
```

Рис. 3.4: Компиляция текста программы

Я компилирую исходный файл hello.asm в obj.o (рис. 3.5).

```
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
```

Рис. 3.5: Компиляция файла и проверка выполненной программы

Передаю объектный файл на обработку компоновщику и делаю проверку (рис. 3.6).

```
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
```

Рис. 3.6: Передача файла на обработку компоновщику

Ввожу команду согласно лабораторной работе. Исполняемый файл имеет имя main, а объектный файл - obj.o (рис. 3.7).

```
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
```

Рис. 3.7: Передача файла на обработку компоновщику

Запускаю исполняемый файл hello (рис. 3.8).

```
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
```

Рис. 3.8: Запуск исполняемого файла

Мне захотелось увидеть формат командной строки LD. Для этого я ввела команду ld -help (рис. 3.9).



```
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ ld --help
Использование ld [параметры] файл...
Параметры:
-a КЛЮЧЕВОЕ СЛОВО                Управление общей библиотекой для совместимости с HP/UX
-A АРХИТЕКТУРА, --architecture АРХИТЕКТУРА    Задать архитектуру
-b ЦЕЛЬ, --format ЦЕЛЬ            Задать цель для следующих входных файлов
-c ФАЙЛ, --mri-script ФАЙЛ        Прочитать сценарий компоновщика в формате MRI
-d, -dc, -dr                      Принудительно делать общие символы определенными
--dependency-file ФАЙЛ Write dependency file    Принудительно удалить членов группы из групп
--force-group-allocation
-e АДРЕС, --entry АДРЕС           Задать начальный адрес
-E, --export-dynamic              Экспортировать все динамические символы
--no-export-dynamic              Отменить действие --export-dynamic
--enable-non-contiguous-regions   Enable support of non-contiguous memory regions
--enable-non-contiguous-regions-warnings        Enable warnings when --enable-non-contiguous-regions may cause unexpected behaviour
-EB                               Компоновать объекты с прямым порядком байтов
-EL                               Компоновать объекты с обратным порядком байтов
-f SHLIB, --auxiliary SHLIB       Вспомогательный фильтр таблицы символов общих объектов
-F SHLIB, --filter SHLIB          Фильтр для таблицы символов общих объектов
-g                               Игнорируется
-G РАЗМЕР, --gpsize РАЗМЕР        Размер маленьких данных (если не указан, то берётся из --shared)
-h ИМЯ_ФАЙЛА, -soname ИМЯ_ФАЙЛА  Задать внутреннее имя общей библиотеки
-I ПРОГРАММА, --dynamic-linker ПРОГРАММА        Назначить ПРОГРАММУ в качестве используемого динамического компоновщика
--no-dynamic-linker              Создать исполняемый файл без заголовка программного интерпретатора
-l LIBNAME, --library LIBNAME     Искать библиотеку с именем LIBNAME
-L КАТАЛОГ, --library-path КАТАЛОГ        Добавить КАТАЛОГ к пути поиска библиотек
--sysroot=<DIRECTORY>            Заменить расположение по умолчанию sysroot
-m ЭМУЛЯЦИЯ                      Задать эмуляцию
-M, --print-map                  Напечатать файл карты на стандартном выводе
```

Рис. 3.9: Формат командной строки

Запускаю на выполнение созданный исполняемый файл (рис. 3.10).

```
dddihmova@dk5n60 ~/work/arch-pc/lab04 $ ./hello
Hello world!
```

Рис. 3.10: Запуск исполняемого файла

## 4 Выполнение заданий для самостоятельной работы

Я создаю копию файла с помощью команды `cp` (рис. 4.1).

```
dddihmova@dk8n62 ~ $ cd work/arch-pc/lab04/  
dddihmova@dk8n62 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
```

Рис. 4.1: Создание копии файла

С помощью текстового редактора `gedit` вношу изменения в текст программы

```
1; lab4.asm  
2 SECTION .data ; Начало секции данных  
3     name: DB 'Dymova Diana',10 ; 'Dymova Diana' плюс  
4 ; символ перевода строки  
5     nameLen: EQU $-name ; Длина строки name  
6  
7 SECTION .text ; Начало секции кода  
8     GLOBAL _start  
9  
10 _start: ; Точка входа в программу  
11     mov eax,4 ; Системный вызов для записи (sys_write)  
12     mov ebx,1 ; Описатель файла '1' - стандартный вывод  
13     mov ecx,name ; Адрес строки name в ехх  
14     mov edx,nameLen ; Размер строки name  
15     int 80h ; Вызов ядра  
16  
17     mov eax,1 ; Системный вызов для выхода (sys_exit)  
18     mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)  
19     int 80h ; Вызов ядра  
20
```

lab4.asm (рис. ??).

Проверяю корректность изменений в файле `lab4.asm`, так чтобы при вводе команды `./name` выводилось моё имя. Также транслирую полученный текст программы в объектный файл и выполняю компоновку объектного файла, запускаю получившийся исполняемый файл (рис. 4.2).

```

dddihmova@dk8n62 ~/work/arch-pc/lab04 $ gedit lab4.asm
dddihmova@dk8n62 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
dddihmova@dk8n62 ~/work/arch-pc/lab04 $ d -m elf_i386 lab4.o -o name
bash: d: команда не найдена
dddihmova@dk8n62 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o name
dddihmova@dk8n62 ~/work/arch-pc/lab04 $ ./name
Dymova Diana

```

Рис. 4.2: Трансляция и компоновка файлов

Я копирую файлы hello.asm и lab4.asm в локальный репозиторий в каталог  
~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/ (рис. 4.3).

```

dddihmova@dk8n62 ~/work/arch-pc/lab04 $ cp hello.asm ~/work/study/2023-2024/Архитектура\ компьютера/arch-
pc/labs/lab04/
dddihmova@dk8n62 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/study/2023-2024/Архитектура\ компьютера/arch-
pc/labs/lab04/

```

Рис. 4.3: Копирование файлов

Загружаю данные на github (рис. 4.4).

```

dddihmova@dk8n62 ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc $ git add .
dddihmova@dk8n62 ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc $ git commit -am 'feat(main): add
files lab-4'
[master 9f622ba] feat(main): add files lab-4
17 files changed, 173 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
rename labs/lab04/report/{report.md => .md} (100%)
create mode 100644 labs/lab04/report/image/1.png
create mode 100644 labs/lab04/report/image/10.png
create mode 100644 labs/lab04/report/image/11.png
create mode 100644 labs/lab04/report/image/12.png
create mode 100644 labs/lab04/report/image/13.png
create mode 100644 labs/lab04/report/image/2.png
create mode 100644 labs/lab04/report/image/3.png
create mode 100644 labs/lab04/report/image/4.png
create mode 100644 labs/lab04/report/image/5.png
create mode 100644 labs/lab04/report/image/6.png
create mode 100644 labs/lab04/report/image/7.png
create mode 100644 labs/lab04/report/image/8.png
create mode 100644 labs/lab04/report/image/9.png
create mode 100644 labs/lab04/report/л04_Дымова_отчет.md
dddihmova@dk8n62 ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc $ git push
Перечисление объектов: 27, готово.
Подсчет объектов: 100% (27/27), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (22/22), готово.
Запись объектов: 100% (22/22), 427.55 КиБ | 4.23 МиБ/с, готово.
Всего 22 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To github.com:ddiina/study20232024.git
 f96d2c1..9f622ba master -> master

```

Рис. 4.4: Загрузка данных

## 5 Выводы

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