Лабораторная работа №4

Чипурной Михаил Евгеньевич

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоить процедуры компиляции и сборки программ, познакомиться с языком ассемблера NASM.

2 Выполнение лабораторной работы

2.1 Программа Hello world!

Создаем каталог для работы с программами на языке ассемблера NASM ([**fig:001**]).

Рисунок 2.1: Создаем каталоги с помощью команды mkdir

Переходим в созданный каталог ([**fig:002**]).

Рисунок 2.2: Переходим в каталог с помощью команды cd

Создаем текстовый файл ([**fig:003**]).

Рисунок 2.3: Создаем текстовый файл hello.asm

Открываем данный файл в текстовом редакторе ([**fig:004**]).

```
; 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
```

Рисунок 2.4: Открываем файл и заполняем его по примеру

2.2 Транаслятор NASM

Преобразуем текст программы в объектный код ([**fig:005**]).

```
mechipurnoyj@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
```

Рисунок 2.5: Используем команду nasm

Проверяем создался ли объектный файл с помощью команды ls ([**fig:006**]).

```
mechipurnoyj@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
```

Рисунок 2.6: Проверяем работу команды

2.3 Расширенный синтаксис командной строки NASM

Компилируем исходный файл ([**fig:007**]).

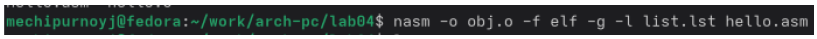

Рисунок 2.7: Преобразуем файл hello.asm в obj.o

Проверяем, как сработала команда ([**fig:008**]).

Рисунок 2.8: Проверяем создание файла командой ls

2.4 Компоновщик LD

Передаем объектный файл на обработку компоновщику ([**fig:009**]).

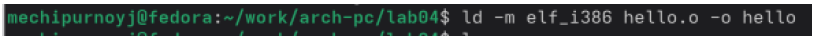

Рисунок 2.9: Используем команду ld

Проверяем создался ли исполняемый файл hello ([**fig:010**]).

Рисунок 2.10: Используем команду ls

Передаем объектный файл на обработку компоновщику ([**fig:011**]).

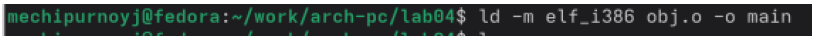

Рисунок 2.11: Используем команду ld, создавая файл main

Проверяем создался ли исполняемый файл hello ([**fig:012**]).

Рисунок 2.12: Используем команду ls

2.5 Запуск исполняемого файла

Запускаем на выполнение созданный исполняемый файл ([**fig:013**]).

```
mechipurnoyj@fedora:~/work/arch-pc/lab04$ ./hello
Hello world!
```

Рисунок 2.13: Используем команду ./hello

2.6 Задание для самостоятельной работы

Создаем копию файла hello.asm ([**fig:014**]).

```
mechipurnoyj@fedora:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
mechipurnoyj@fedora:~/work/arch-pc/lab04$
```

Рисунок 2.14: Используем команду cp

Открываем файл и редактируем его ([**fig:015**]).

```
mechipurnoyj@fedora:~/work/arch-pc/lab04$ gedit lab4.asm
```

Рисунок 2.15: Открываем файл в текстовом редакторе

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

Рисунок 2.16: Редактируем файл для своего имени и фамилии

Прописывем те же команды, что и с первой программой ([**fig:017**]).

```
mechipurnoyj@fedora:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
mechipurnoyj@fedora:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
mechipurnoyj@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o hello
mechipurnoyj@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
mechipurnoyj@fedora:~/work/arch-pc/lab04$ ./hello
Чипурной Михаил
mechipurnoyj@fedora:~/work/arch-pc/lab04$
```

Рисунок 2.17: Прописываем команды для работы файла и запускаем программу

Копируем файлы в локальный репозиторий ([**fig:018**]).

```
mechipurnoyj@fedora:~/work/arch-pc/lab04$ cp hello.asm ~/work/study/2025-2026/"Архитектура
компьютера"/arch-pc/labs/lab04/
mechipurnoyj@fedora:~/work/arch-pc/lab04$ cp lab4.asm ~/work/study/2025-2026/"Архитектура к
омпьютера"/arch-pc/labs/lab04/
mechipurnoyj@fedora:~/work/arch-pc/lab04$
```

Рисунок 2.18: Копируем файлы в каталог с ЛР4

Переходим в каталог лабораторных работ и загружаем файлы на Github ([**fig:019**]).

```
mechipurnoyj@fedora:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git add .
mechipurnoyj@fedora:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): add files lab-4'
[master d35beda] feat(main): add files lab-4
 19 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 labs/lab04/report/image/рис1.jpg
 create mode 100644 labs/lab04/report/image/рис10.jpg
 create mode 100644 labs/lab04/report/image/рис11.png
 create mode 100644 labs/lab04/report/image/рис12.jpg
 create mode 100644 labs/lab04/report/image/рис13.jpg
 create mode 100644 labs/lab04/report/image/рис14.jpg
 create mode 100644 labs/lab04/report/image/рис15.png
 create mode 100644 labs/lab04/report/image/рис16.jpg
 create mode 100644 labs/lab04/report/image/рис17.jpg
 create mode 100644 labs/lab04/report/image/рис18.jpg
 create mode 100644 labs/lab04/report/image/рис19.jpg
 create mode 100644 labs/lab04/report/image/рис2.png
 create mode 100644 labs/lab04/report/image/рис3.jpg
 create mode 100644 labs/lab04/report/image/рис4.jpg
 create mode 100644 labs/lab04/report/image/рис5.png
 create mode 100644 labs/lab04/report/image/рис6.jpg
 create mode 100644 labs/lab04/report/image/рис7.png
 create mode 100644 labs/lab04/report/image/рис8.jpg
mechipurnoyj@fedora:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 41, готово.
Подсчет объектов: 100% (41/41), готово.
При сжатии изменений используется до 10 потоков
Сжатие объектов: 100% (34/34), готово.
Запись объектов: 100% (35/35), 851.41 КиБ | 7.47 МиБ/с, готово.
Total 35 (delta 8), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (8/8), completed with 3 local objects.
To github.com:ChipurnoyM/study_2025-2026_arh-pc.git
   dfa9aa5..d35beda  master -> master
mechipurnoyj@fedora:~/work/study/2025-2026/Архитектура компьютера/arch-pc$
```

Рисунок 2.19: Загружаем файлы

3 Выводы

В ходе лабораторной работы мы изучили основы работы с ассемблером NASM в операционной системе Linux. Мы приобрели практические навыки написания, трансляции и выполнения низкоуровневых программ на языке ассемблера.