Отчёт по лабораторной работе №3

Дисциплина: Архитектура Компьютера

Азарцова Вероника Валерьевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Получение практических навыков по выполнению процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

1. Ознакомление с теоретическим введением.
2. Создание программы.
3. Транслирование и компоновка.
4. Запуск полученного файла.
5. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

Основными функциональными элементами любой ЭВМ являются центральный процессор, память и периферийные устройства. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера.

Регистры — сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций, которые делятся на два типа: регистры общего назначения и специальные регистры.

Большинство команд в программах написанных на ассемблере используют регистры в качестве операндов. 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.

Названия основных регистров общего назначения:

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 — 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.

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 (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.

4 Выполнение лабораторной работы

4.1 Программа Hello World!

Устанавливаю NASM (рис. 4.1).

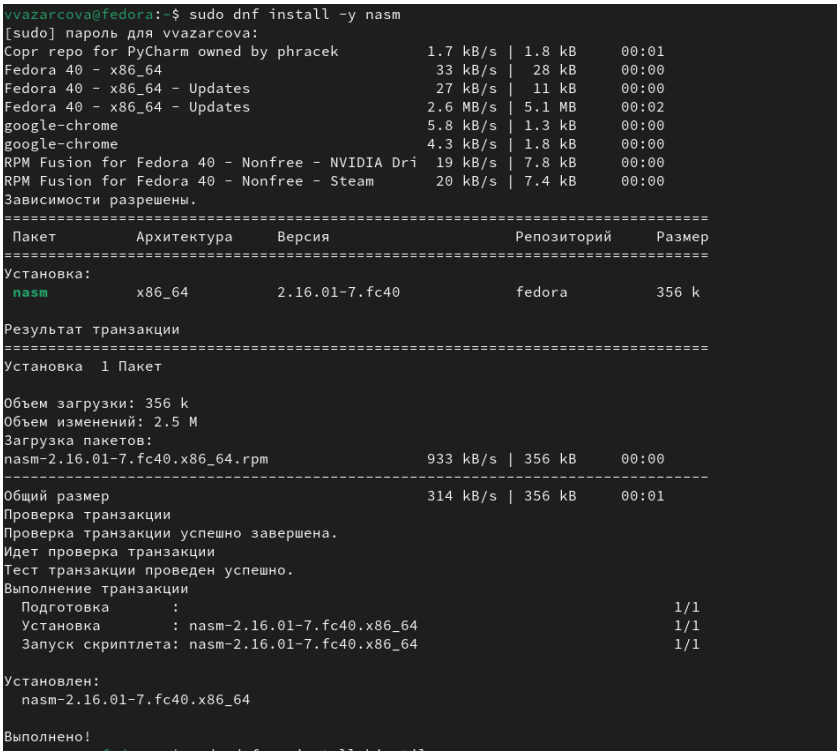

Рис. 4.1: Установка NASM

Создаю каталог для работы с программами на языке ассемблера NASM (рис. 4.2).

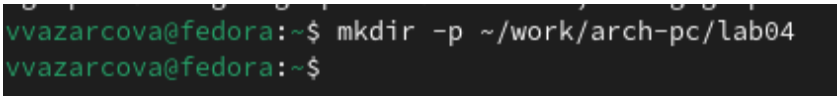

Рис. 4.2: Создание каталога для работы

Перехожу в созданный каталог (рис. 4.3).

Рис. 4.3: Переход в каталог

Создаю текстовый файл с именем hello.asm (рис. 4.4).

Рис. 4.4: Создание hello.asm

Открываю файл с помощью текстового редактора gedit (рис. 4.5).

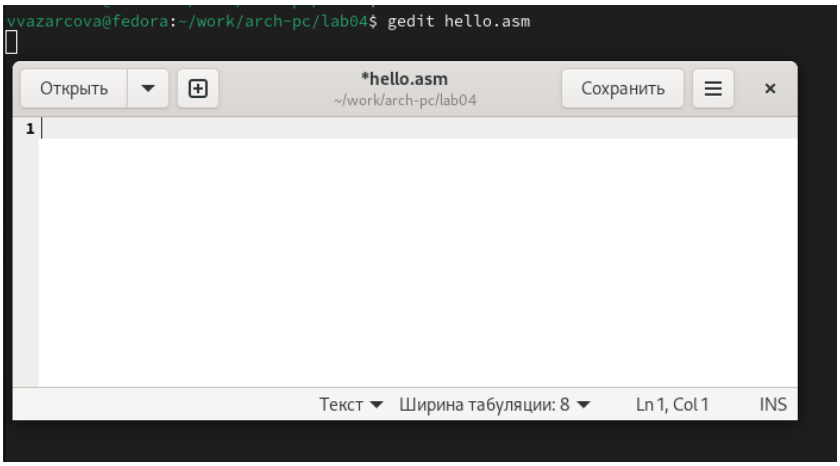

Рис. 4.5: Файл открытый в gedit

Ввожу текст программы в текстовый файл (рис. 4.6).

Рис. 4.6: Ввод текста программы

4.2 Транслятор NASM

Компилирую написанный ранее текст программы в обьектный код (рис. 4.7).

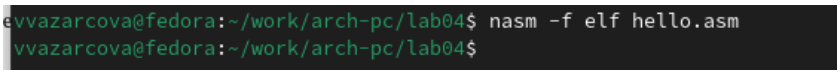

Рис. 4.7: Компилирование

Проверяю, что транслятор преобразовал текст программы из hello.asm в hello.o. Преобразование произошло, следовательно, текст программы набран правильно. (рис. 4.8).

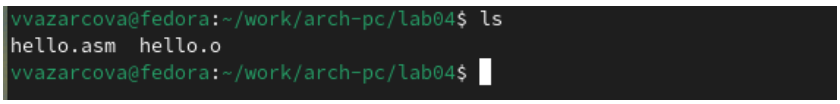

Рис. 4.8: Проверка преобразования

С помощью опций команды nasm компилирую исходный файл hello.asm в obj.o с созданием файла листинга list.lst (рис. 4.9).

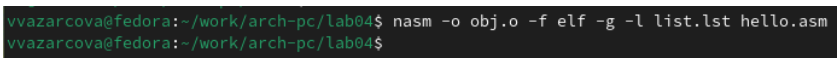

Рис. 4.9: Компилирование с опциями

С помощью команды ls проверяю, что файлы были созданы. Всё создано верно. (рис. 4.10).

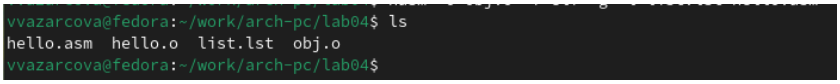

Рис. 4.10: Проверка компиляции

Обрабатываю обьектный файл компановщиком чтобы получить исполняемую программу (рис. 4.11).

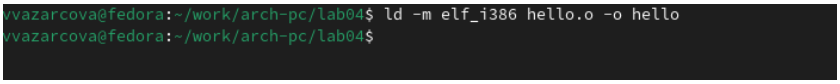

Рис. 4.11: Компановка

С помощью команды ls проверяю, что исполняемый файл hello был создан (рис. 4.12).

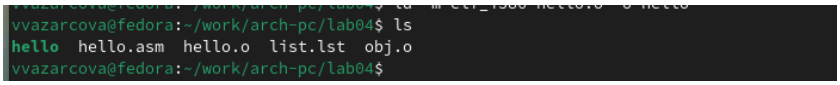

Рис. 4.12: Проверка создания исполняемого файла

Выполняю команду “ld -m elf_i386 obj.o -o main”. При её выполнении исходный обьектный файл это obj.o, а полученный исполняемый файл - main (рис. 4.13).

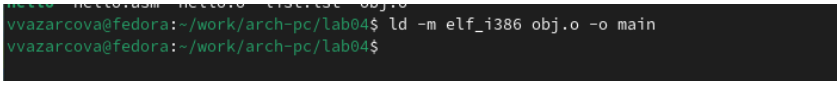

Рис. 4.13: Компановка с опциями

Проверяю, что файл main был создан (рис. 4.14).

Рис. 4.14: Проверка создания файла main

4.3 Запуск

Запускаю на выполнение созданный исполняемый файл. Вижу, что вывелся нужный текст (рис. 4.15).

Рис. 4.15: Запуск исполняемого файла

4.4 Задание для самостоятельной работы

1. В каталоге ~/work/arch-pc/lab04 с помощью команды cp создаю копию файла hello.asm с именем lab4.asm (рис. 4.16).

Рис. 4.16: Копирование файла

С помощью ls проверяю, что файл был скопирован успешно (рис. 4.17).

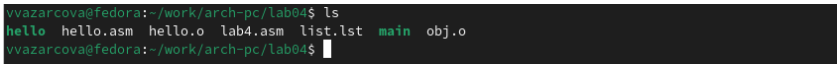

Рис. 4.17: Проверка копирования файла

2. Открываю файл с помощью gedit (рис. 4.18).

Рис. 4.18: Открытие файла в gedit

Измененяю текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилось “Азарцова Вероника” (рис. 4.19).

Рис. 4.19: Редактирование текста программы

3. Транслирую полученный текст программы lab4.asm в объектный файл (рис. 4.20).

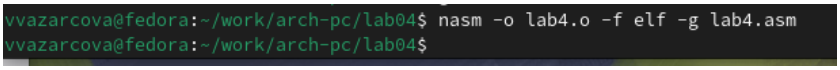

Рис. 4.20: Транслирование lab4.asm

Проверяю наличие оттранслированного обьектного файла с помощью ls. (рис. 4.21).

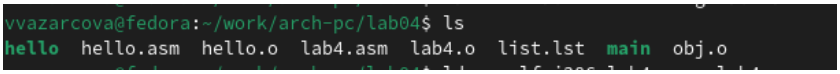

Рис. 4.21: Проверка наличия оттранслированного файла

Выполняю компоновку объектного файла (рис. 4.22).

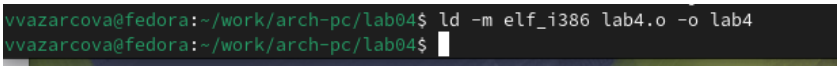

Рис. 4.22: Компановка обьектного файла

Проверяю успешность компановки файла с помощью ls (рис. 4.23).

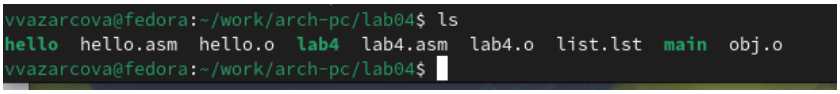

Рис. 4.23: Проверка наличия скомпанованного файла

Запускаю получившийся исполняемый файл (рис. 4.24).

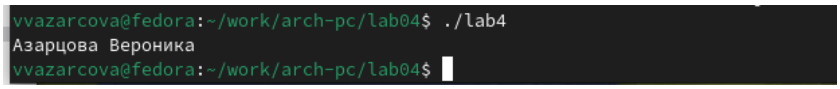

Рис. 4.24: Запуск исполняемого файла

4. Копирую файл hello.asm в мой локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/ (рис. 4.25).

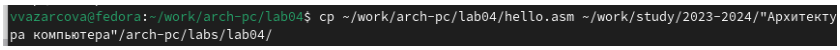

Рис. 4.25: Копирование hello.asm

Аналогично копирую lab4.asm (рис. 4.26).

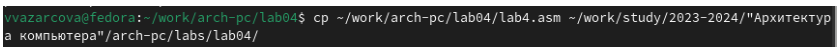

Рис. 4.26: Копирование lab4.asm

Проверяю наличие скопированных файлов в нужном каталоге с помощью ls (рис. 4.27).

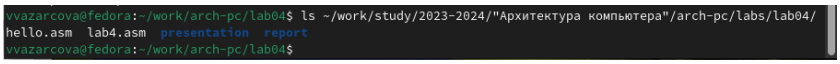

Рис. 4.27: Проверка наличия скопированных файлов

Загружаю файлы на Github.

5 Выводы

Подводя итоги проведённой лабораторной работе, я получила практические навыки по выполнению процедуры компиляции и сборки программ, написанных на ассемблере NASM.

Список литературы