Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра безопасности информационных систем (БИС)

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕР

Отчет по лабораторной работе №2

по дисциплине «Системное программирование»

Студентка гр.737-1

\_\_\_\_\_\_ Агеева В.С.

\_\_.\_\_.2021г

Принял

Руководитель

доцент кафедры БИС

\_\_\_\_\_\_\_ Романов А.С.

\_\_.\_\_.2021г

Томск 2021

1 Введение

Цель работы: познакомиться со структурой программы на языке Ассемблер, разновидностями и назначением сегментов, способами организации простых и сложных типов данных, познакомиться со средствами создания программ на Ассемблере для ОС Linux.

Задание на лабораторную работу:

1. Задача: дан массив из 10 беззнаковых слов. Инвертировать биты старших байтов всех элементов массива. Найти сумму четных элементов полученного массива.

2 Ход работы

Для начала выполнения лабораторной работы необходимо сначала обновить ubuntu и установить расширенный ассемблер nasm (рисунок 2.1).

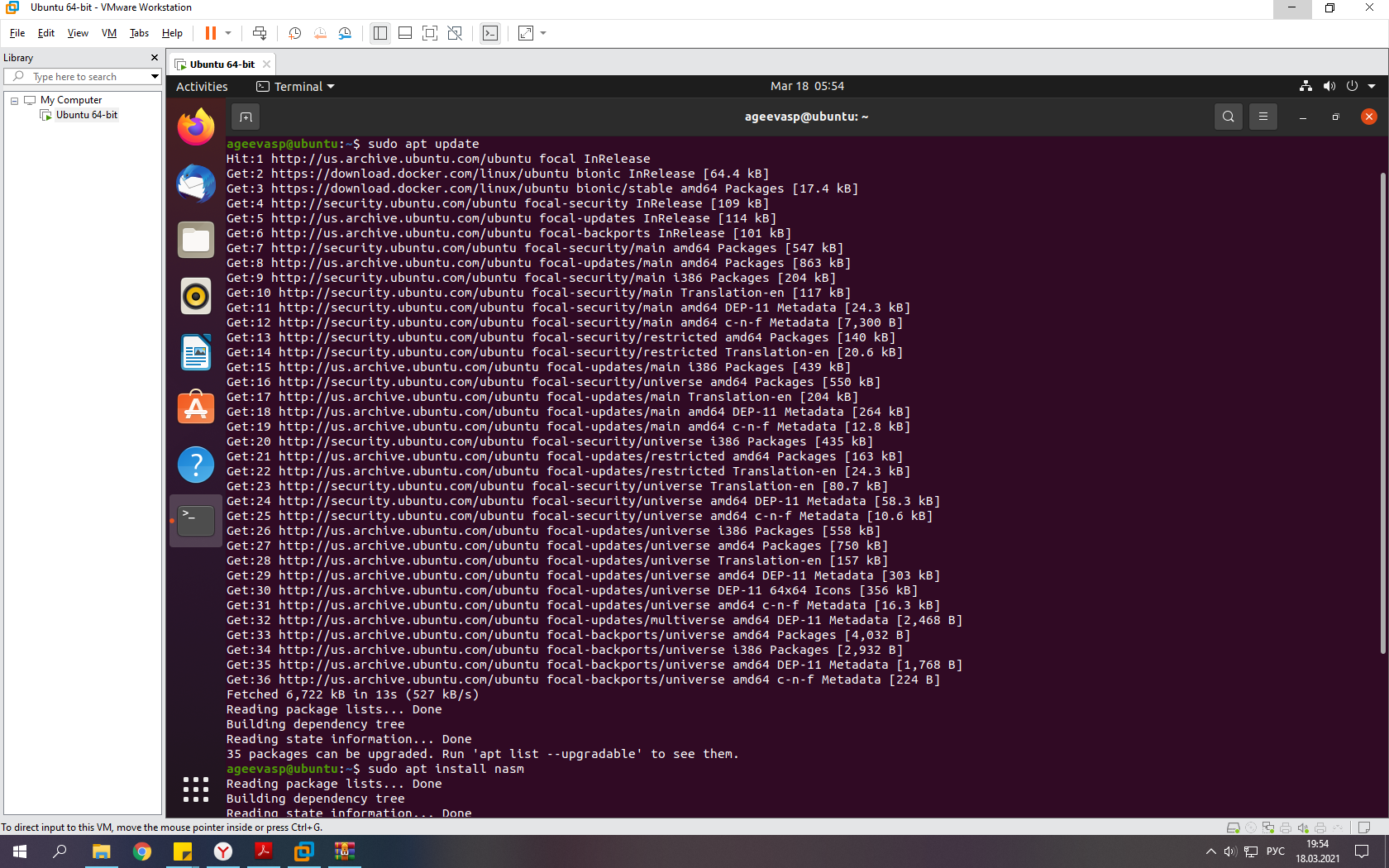


Рисунок 2.1 – Обновление ubuntu и установка nasm

Далее через nano создадим файл, куда запишем код ассемблера (рисунок 2.2).

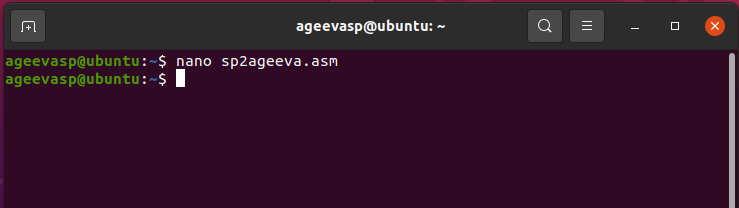


Рисунок 2.2 - Создание файла

На рисунке 2.3 показан код ассемблера.

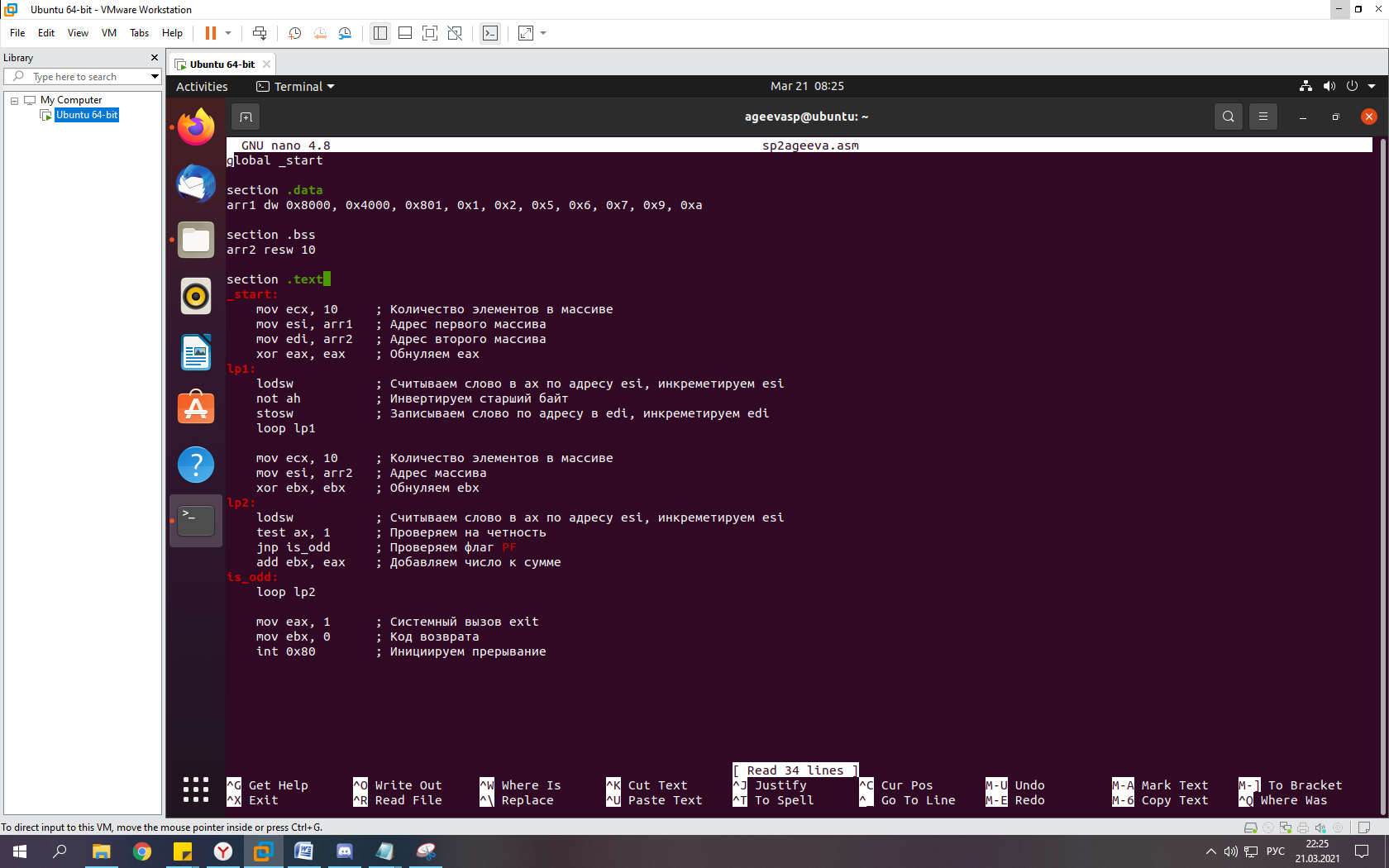


Рисунок 2.3 – Код по варианту на ассемблере

Далее командой nasm –f elf sp2ageeva.asm скомпилируем наш код, а командой ls проверим, чтобы появился объектный файл sp2ageeva.o (рисунок 2.4).

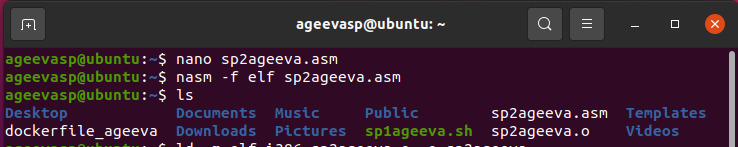


Рисунок 2.4 – Компилирование файла и создание объектного файла

Далее скачаем и установим в системе компилятор gcc (рисунок 2.5).



Рисунок 2.5 – Скачивание и установка компилятора gcc

Командой ld –m elf\_i386 sp2ageeva.o –o sp2ageeva происходит связывание. Также для выполнения лабораторной работы нужно скачать и установить отладчик gdb (рисунок 2.6).

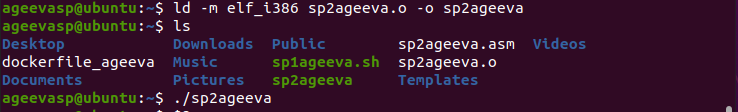


Рисунок 2.6 – Связывание объектного файла с исполняемым файлом и

установка отладчика gdb

После того как скачали отладчик, запустим его и проверим работоспособность программы (рисунки 2.7 – 2.11).

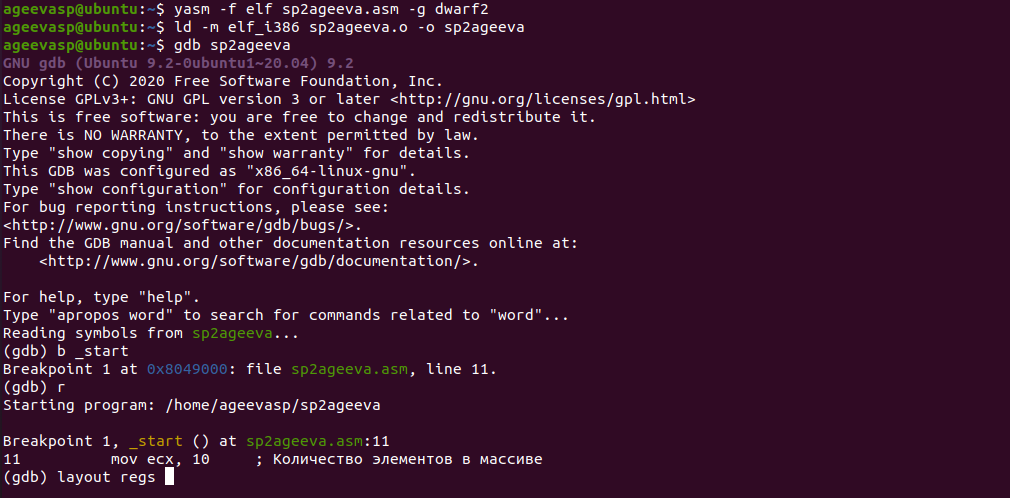


Рисунок 2.7 – Запуск компилятора

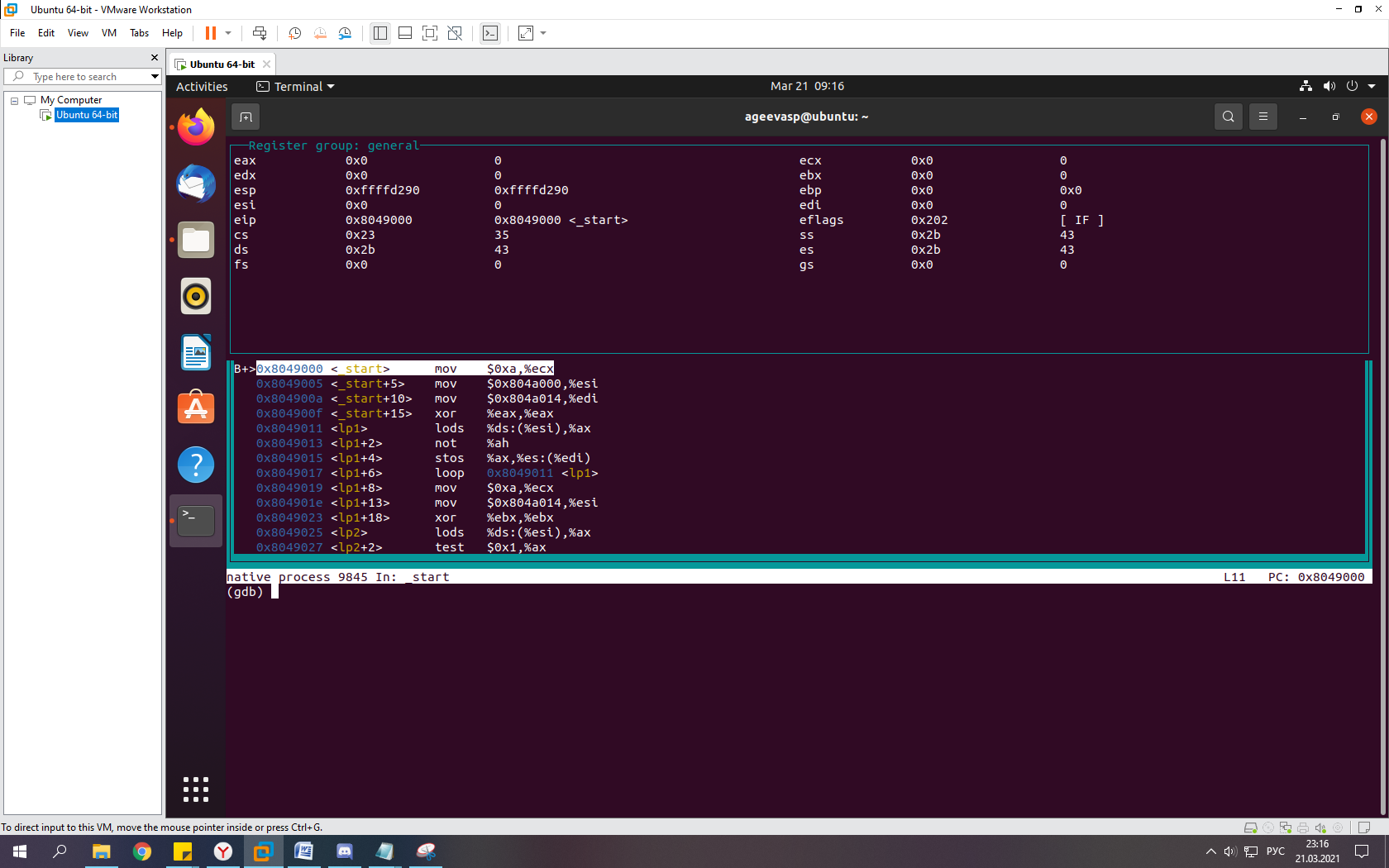


Рисунок 2.8 – Запущенный компилятор

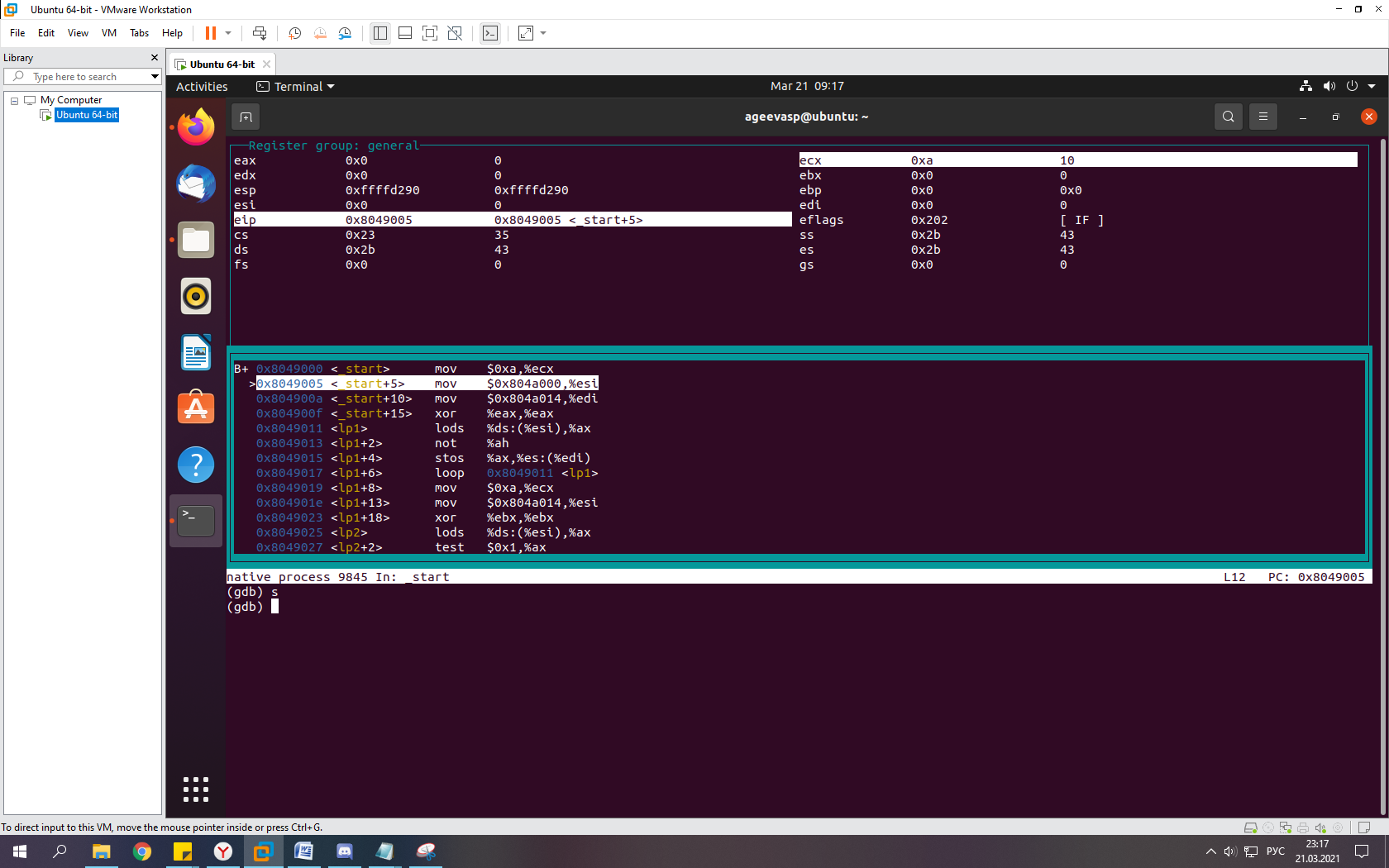


Рисунок 2.9 – Начала работы компилятора

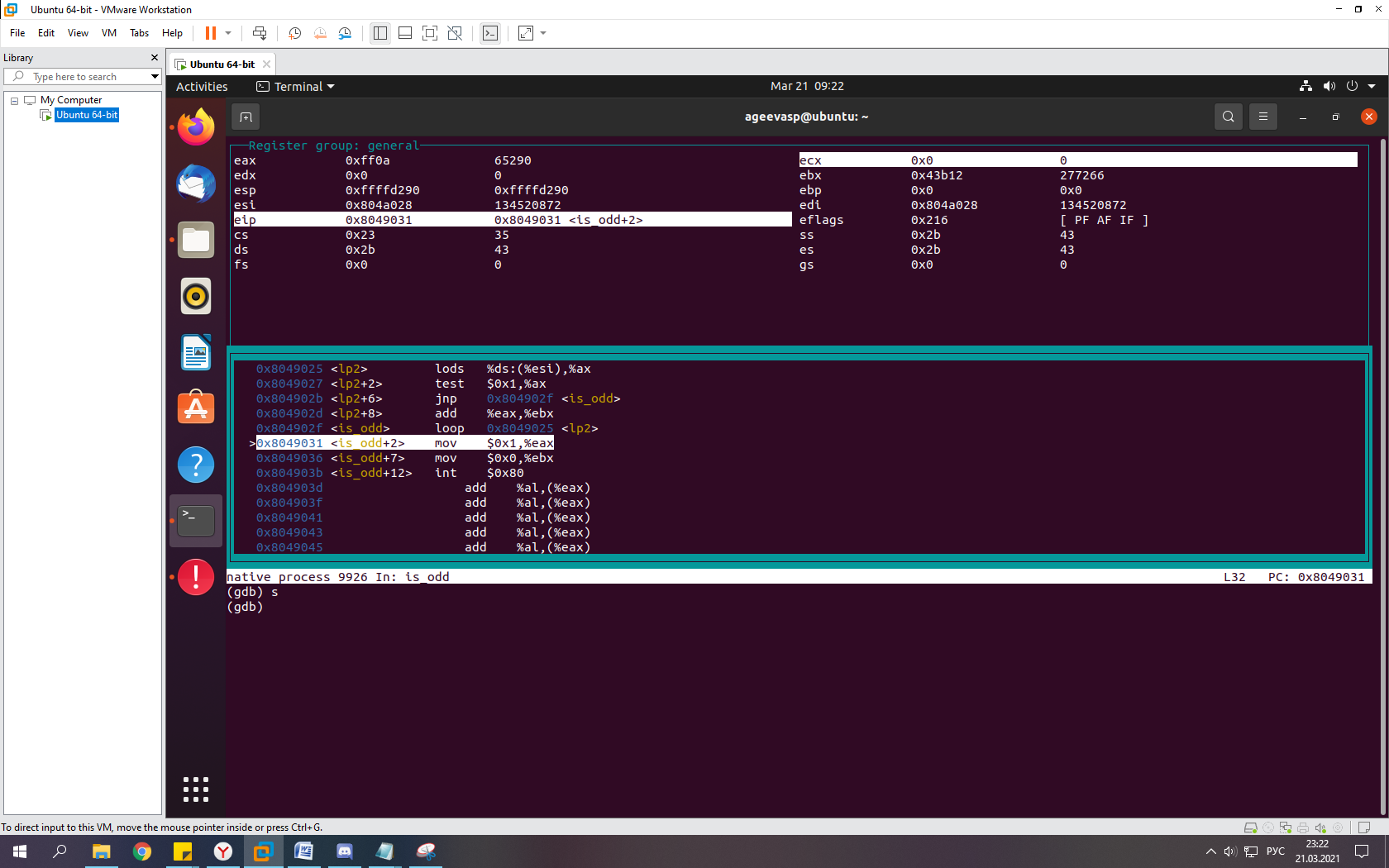


Рисунок 2.10 – Завершение работы компилятора с результатом суммы 43b12

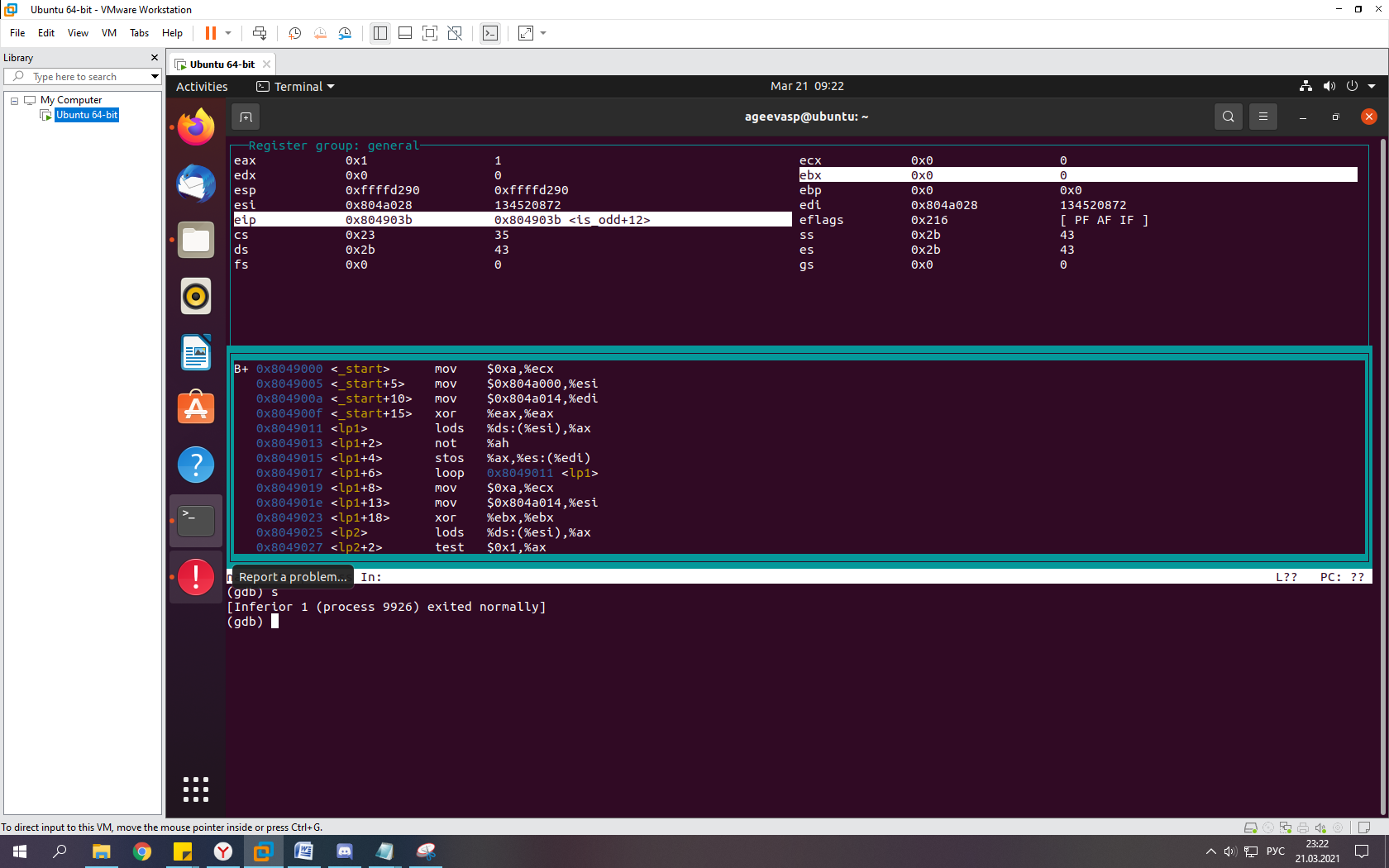


Рисунок 2.11 – Завершение полностью работы компилятора

После того, как мы проверили работоспособность программы через компилятор, нужно дизассемблировать код, для этого нужно ввести команду: objdump -d -M i386 -M intel-mnemonic sp2ageeva (рисунок 2.12).

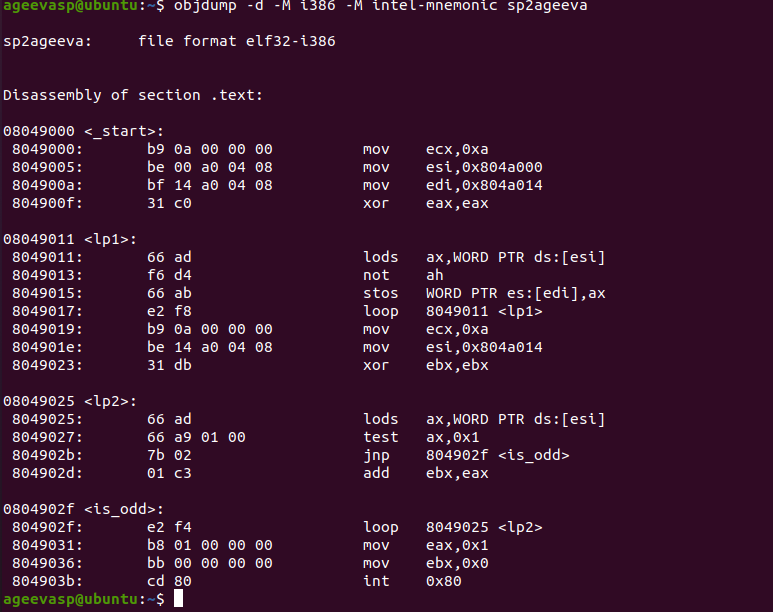


Рисунок 2.12 – Дизассемблирование кода

Из рисунка видно следующую информацию:

- адреса;

- машинный код;

- мнемоника;

- операнды.

После того как сделали всю работу на ассемблере, необходимо написать программу на языке С, запустить через компилятор gcc и дизассемблировать. В итоге, сравнить какой код выполняется быстрее (рисунки 2.13 – 2.15).

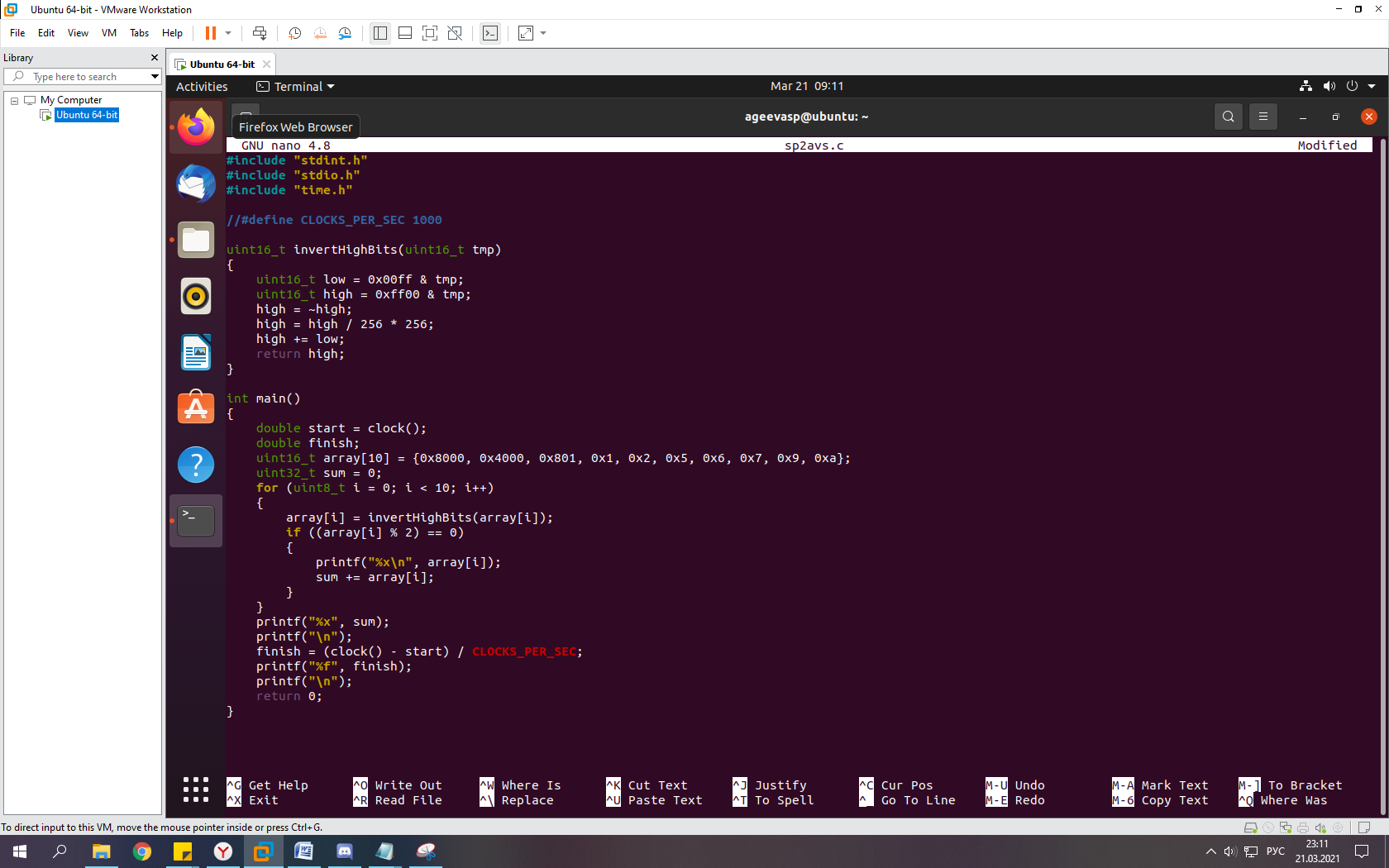


Рисунок 2.13 – Код на языке С

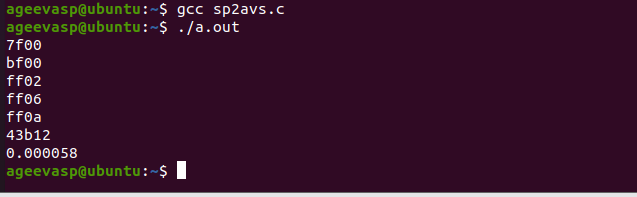
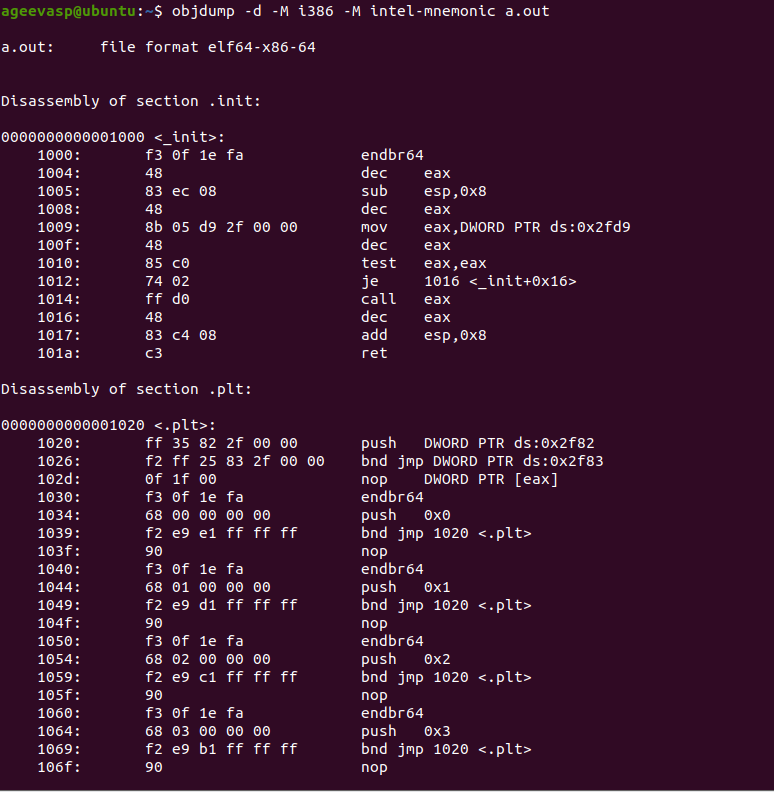


Рисунок 2.14 – Результат программы



…

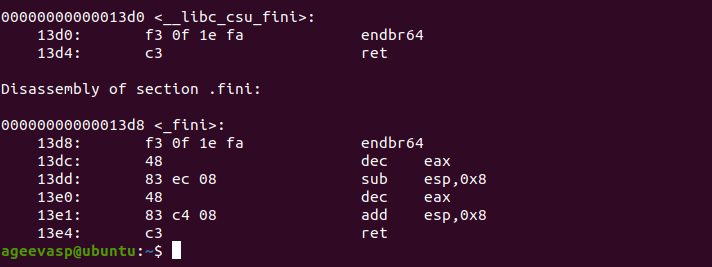


Рисунок 2.15 – Дизассемблированный код высокого языка программирования

3 Заключение

В ходе работы были написаны программы на языке высокого уровня и ассемблера, выполняющие одинаковый функционал.

В ходе анализа дизассемблированного кода было обнаружено, что полученный код высокоуровневой программы многократно (в шесть раз) превосходит по объему аналогичный код программы, написанной на ассемблере. Так же были получены навыки отладки программ с помощью отладчика gdb.

Ссылка на github: https://github.com/7371avs/SP/tree/main