Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

“ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ” (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕР

Отчёт по лабораторной работе №2

по дисциплине «Системное программирование»

Студент гр.748

\_\_\_\_\_\_\_\_Е.В. Граборова

«15» марта 2022

Принял

М.н.с ИСИБ

\_\_\_\_\_\_\_\_Е. О. Калинин

«\_\_» \_\_\_\_\_\_ 2022

Томск 2022

# 1 Введение

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

Задание:

* подготовить образ операционной системы Linux для Docker. Установить компилятор GCC и другие необходимые пакеты. Скомпилировать простейшую программу на Ассемблере с помощью компилятора GCC;
* воспользоваться отладчиком GDB и научиться пользоваться представляемой им информацией;
* получить индивидуальное задание у преподавателя и реализовать соответствующую программу на Ассемблере и на языке высокого уровня.

Дизассемблировать обе программы, провести сравнительный анализ скорости работы программ, объема полученного дизассемблированного кода, попробовать оптимизировать программы. Сделать выводы.

Вариант 6.

Задача: дан массив из 10 беззнаковых чисел (байтов). Коды четных элементов массива сдвинуть логически влево, а нечетных – арифметически вправо.

# 2 Ход работы

Перед началом работы необходимо обновить компиляторы и все дополнительные компоненты для успешной компиляции программ (рисунок 1)

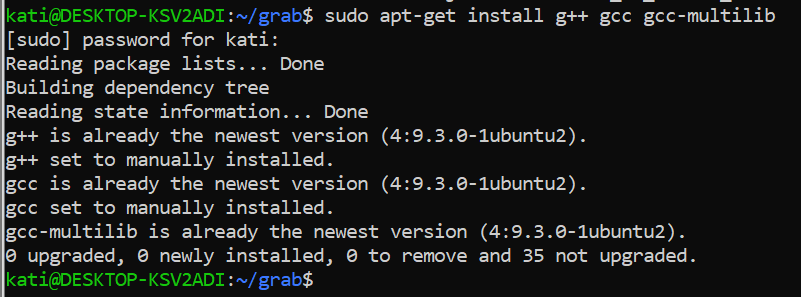


Рисунок 1 – Обновление компиляторов

Дальше необходимо написать первую простую программу на языке assembler, которая выводит на экран «Hello world», скомпилировать и запустить ее (рисунок 2,3).

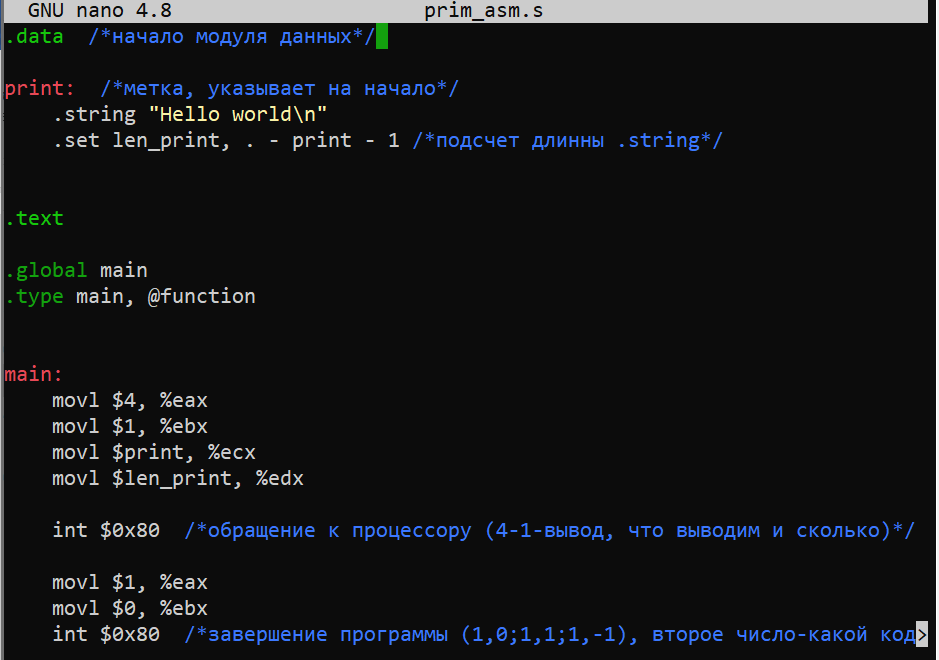


Рисунок 2 – Программа-приветствие

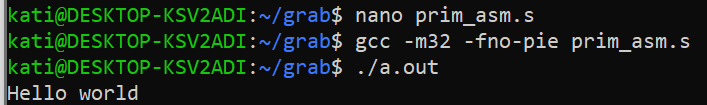


Рисунок 3 – Компиляция и запуск

Для того, чтобы воспользоваться отладчиком, необходимо предварительно скомпилировать программу с флагом «-g» для добавления в программу отладочной информации. После можно пользоваться отладчиком GDB (рисунок 4). Для начала отладчику необходимо указать точку остановки в программе (break), дальше запустить выполнение программы до этой самой точки (run), а затем уже можно покомандно выполнять инструкции (next, step).

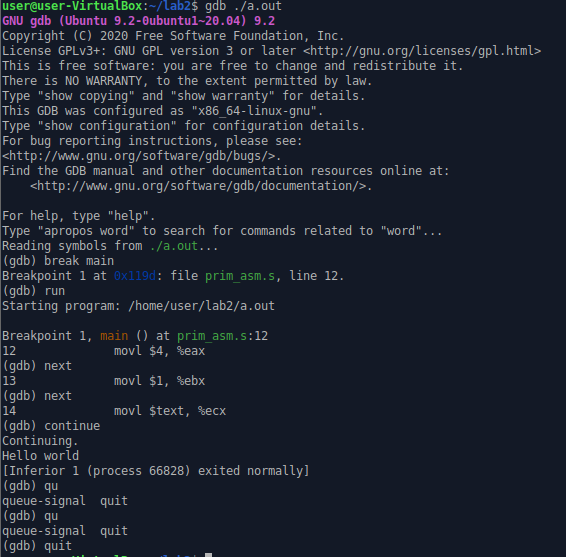


Рисунок 4 – Работа с отладчиком GDB

Перейдя дальше по списку, была написана на языке assembler, скомпилирована, запущена и проверена на правильность программа согласно варианту (рисунок 5).

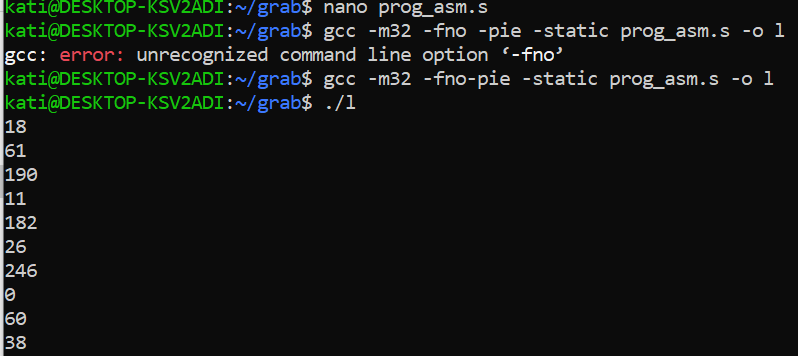


Рисунок 5 – Программа на assembler

Точно также была написана и проверена программа, написанная на языке высокого уровня C++ (рисунок 6,7).

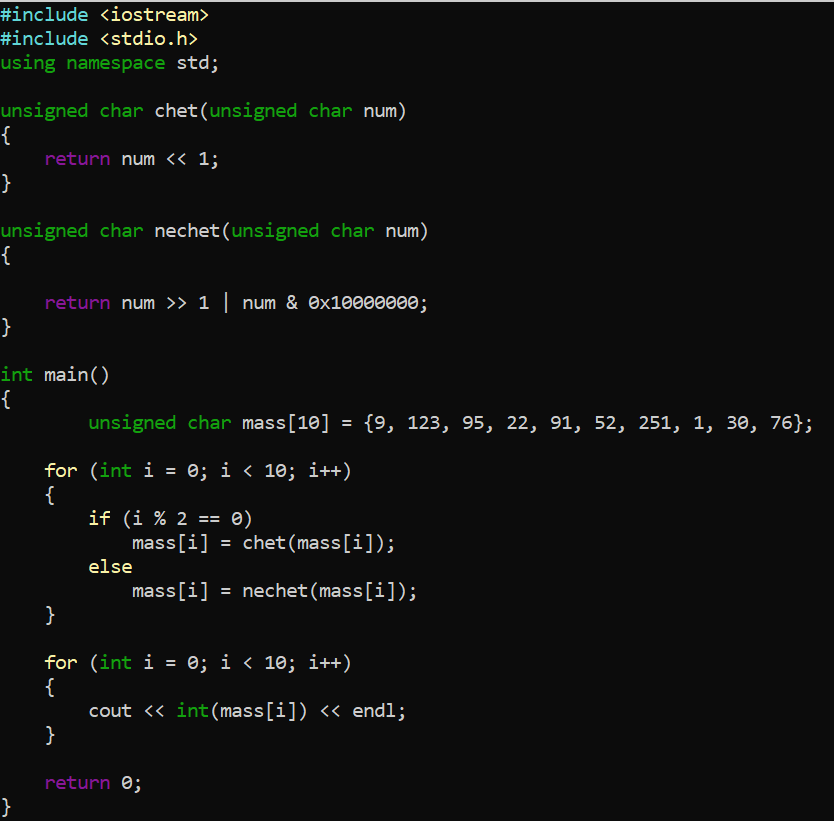


Рисунок 6 – Программа на языке высокого уровня

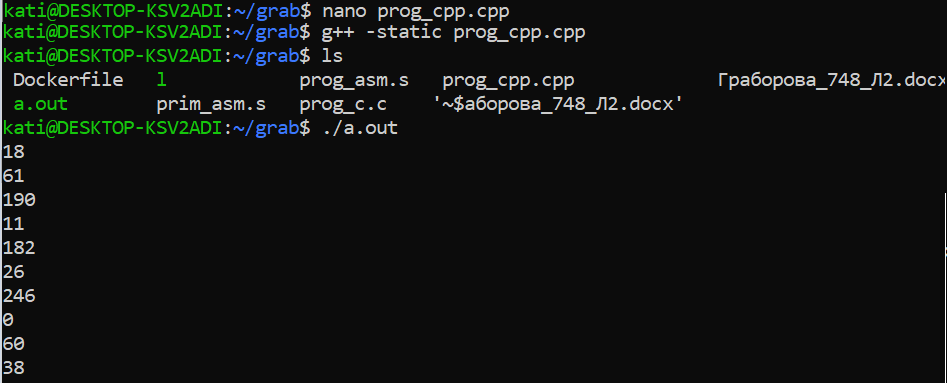


Рисунок 7 – Компиляция и запуск программы

При проведении сравнения результатов дизассемблирования кодов на assembler (рисунок 7) и на C++ (рисунок 8) было замечено, что у языка высокого уровня код менее читаемый и куда более больший по объему.

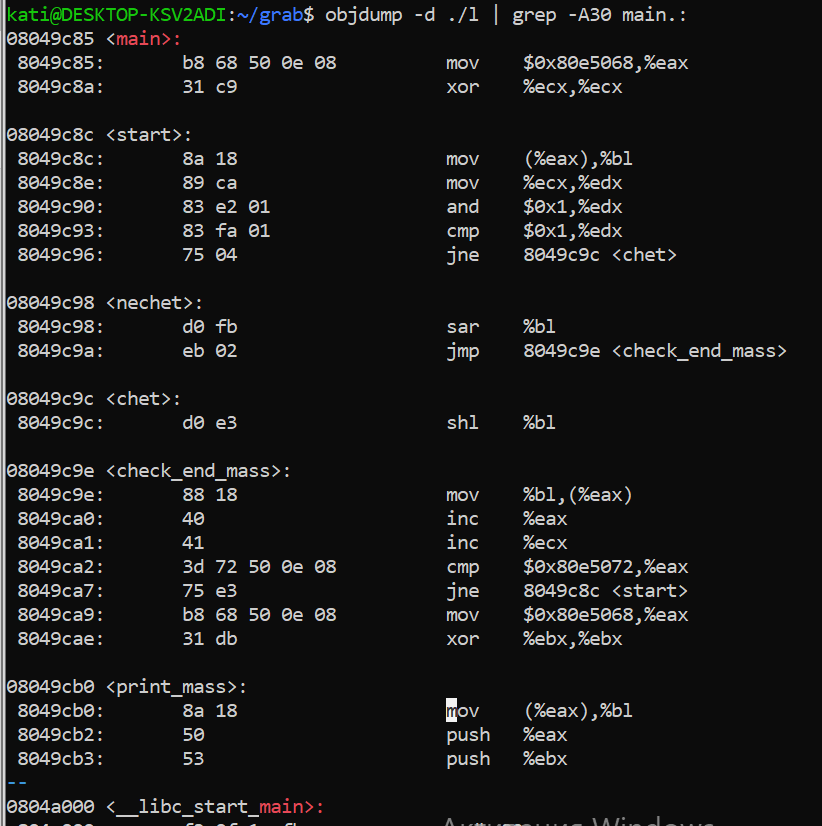


Рисунок 7 – Дизассемблирование первой программы

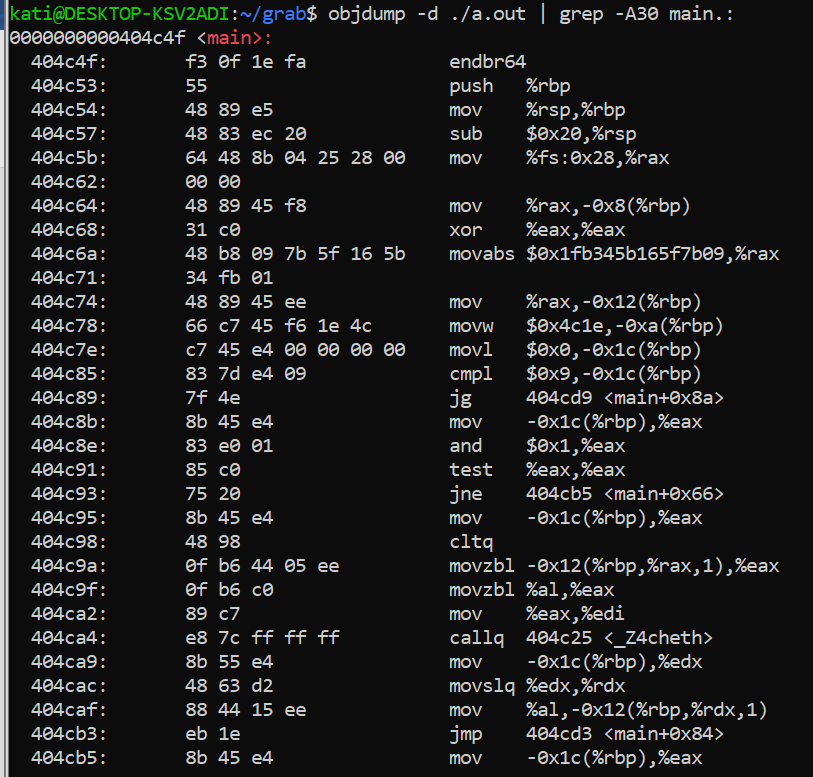
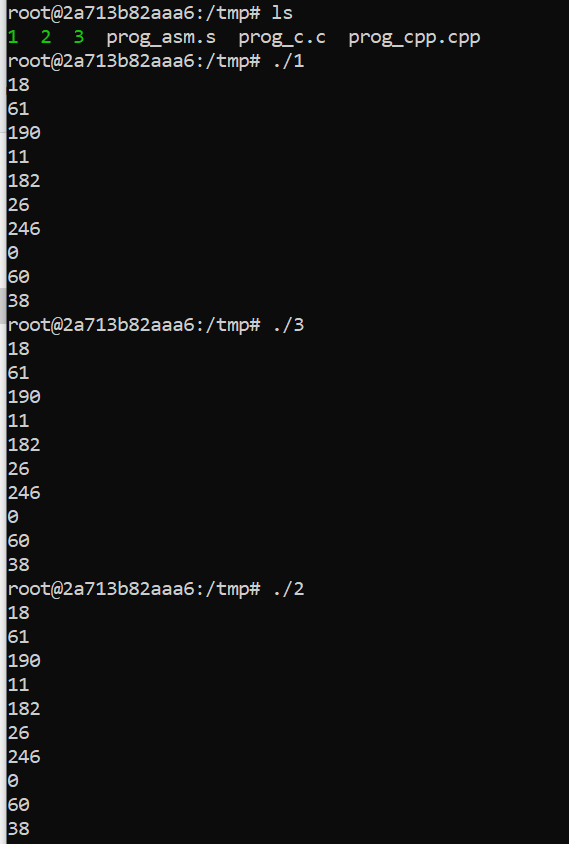


Рисунок 8 – Дизассемблирование второй программы

Если сравнить итоговые размеры исполняемых файлов (рисунок 8), то код, написанный на assembler, будет почти в 4 раза меньше занимать объема, чем программа на С++, хоть и результат их работы одинаковый.



Рисунок 8 – Сравнение размеров программ



Докерфайл работает

Для сокращения программ можно воспользоваться командой «strip» (рисунок 9). Она незначительно, но все-таки уменьшила итоговый размер программ.

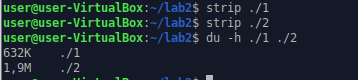


Рисунок 9 – Команда strip

Можно переписать программу с С++ на С и применить команду «strip» (рисунок 10). Это не составит большого труда, так как программы достаточно простые и будут отличаться лишь подключаемой библиотекой вывода.

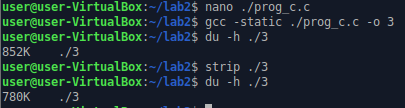


Рисунок 10 – Программа на С

По результату проведенной работы можно сравнить размер программ (рисунок 11) и их время выполнения (рисунок 12). Самым малым размером обладает программа на языке assembler, хотя и на С лишь немного больше ее. По времени выполнения они практически неразличимы, потому что программы достаточно просты и выполняются очень быстро.



Рисунок 11 – Размеры программ

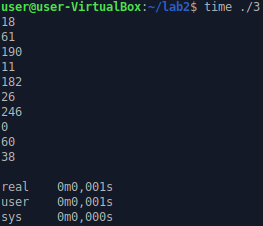
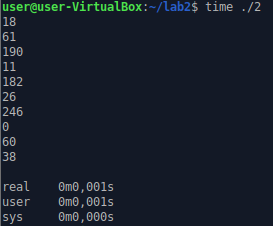
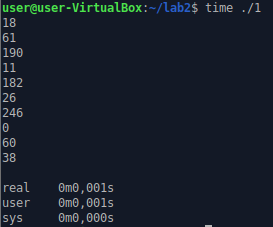


Рисунок 12 – Время выполнения программ

Последним шагом осталось написание Dockerfile, создание на его основе образ (рисунок 13), запуск контейнера и проверка правильности работы (рисунок 14).

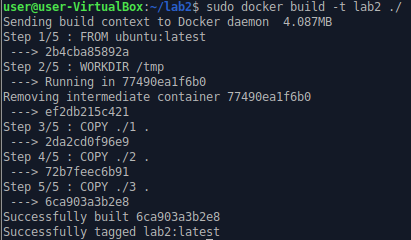


Рисунок 13 – Создание docker-образа

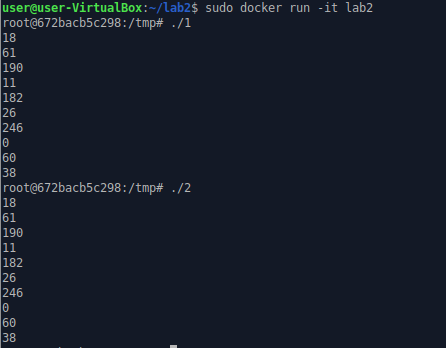


Рисунок 14 – Запуск контейнера

# 3 Заключение

В результате выполнения данной лабораторной работы было произведено ознакомление со структурой программы на языке Ассемблер, разновидностями и назначением сегментов, способами организации простых и сложных типов данных, изучение форматов и правил работы с синтаксисами AT&T, ознакомление с возможностями GCC для работы с Ассемблером и средствами создания программ на Ассемблере для ОС Linux.

# Приложение А

# Репозиторий на GitHub

.data

mass:

.byte 9, 123, 95, 22, 91, 52, 251, 1, 30, 76

mass\_end:

print\_format:

.string "%d\n"

.text

.global main

.type main, @function

main:

movl $mass, %eax

xorl %ecx, %ecx

start:

mov (%eax), %bl

movl %ecx, %edx

and $1, %edx

cmpl $1, %edx

jne chet

nechet:

sar $1, %bl

jmp check\_end\_mass

chet:

shl $1, %bl

check\_end\_mass:

movb %bl, (%eax)

inc %eax

inc %ecx

cmpl $mass\_end, %eax

jne start

movl $mass, %eax

xorl %ebx, %ebx

print\_mass:

movb (%eax), %bl

pushl %eax

pushl %ebx

pushl $print\_format

call printf

addl $8, %esp

popl %eax

inc %eax

cmpl $mass\_end, %eax

jne print\_mass

movl $1, %eax

movl $0, %ebx

int $0x80