

Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ
И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра комплексной информационной безопасности электронно-вычис-
лительных систем (КИБЭВС)

КОМБИНИРОВАННЫЕ ПРОГРАММЫ. СВЯЗЫВАНИЕ РАЗНОЯЗЫКОВЫХ
МОДУЛЕЙ

Отчет по лабораторной работе №3
по дисциплине «Системное программирование»

Выполнил:

Студент гр. 718

_____ Сахарбеков Р.Д.

____. ____2022

Принял:

М.н.с., ИСИБ

_____ Калинин Е.О.

____. ____2022

Введение

Познакомиться с основными способами передачи параметров подпрограмм, особенностями передачи управления между модулями, научиться писать комбинированные программы, в которых модули Ассемблера вызываются из модулей, написанных на высокоуровневых языках программирования.

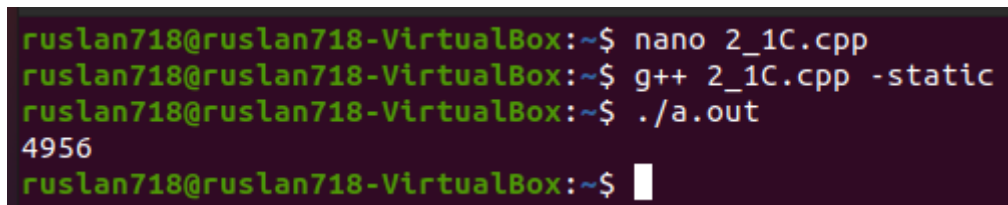
Вариант 1:

Напишите программу, в которой создается два числовых массива одинакового размера. Необходимо вычислить сумму попарных произведений элементов этих массивов. Так, если через a_k и b_k , обозначить элементы массивов (индекс $0 \leq k < n$), то необходимо вычислить сумму.

2 ХОД РАБОТЫ

2.1 НАПИСАНИЕ КОДА

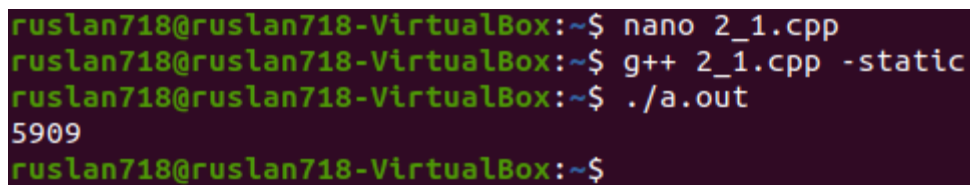
Программа была успешно откомпилирована и запущена (рисунок 2.1). Код программы представлен в приложении А.



```
ruslan718@ruslan718-VirtualBox:~$ nano 2_1C.cpp
ruslan718@ruslan718-VirtualBox:~$ g++ 2_1C.cpp -static
ruslan718@ruslan718-VirtualBox:~$ ./a.out
4956
ruslan718@ruslan718-VirtualBox:~$
```

Рисунок 2.1 – Компиляция и запуск программы (1)

Далее часть кода с выводом элементов была заменена на ассемблерную вставку, а так же код был разбит на функции. Код был успешно скомпилирован и запущен (рисунок 2.2). Код программы представлен в приложении Б.



```
ruslan718@ruslan718-VirtualBox:~$ nano 2_1.cpp
ruslan718@ruslan718-VirtualBox:~$ g++ 2_1.cpp -static
ruslan718@ruslan718-VirtualBox:~$ ./a.out
5909
ruslan718@ruslan718-VirtualBox:~$
```

Рисунок 2.2 – Компиляция и запуск программы (2)

2.2 РАБОТА С DOCKERFILE

Далее был создан Dockerfile. После этого был создан (рисунок 2.3) и запущен (рисунок 2.4) образ на основе Dockerfile. Код Dockerfile представлен в приложении В.

```
ruslan718@ruslan718-VirtualBox:~$ nano Dockerfile
ruslan718@ruslan718-VirtualBox:~$ docker build -t test .
Sending build context to Docker daemon 40.17MB
Step 1/7 : FROM ubuntu
--> 54c9d81cbb44
Step 2/7 : COPY 2_1.cpp .
--> Using cache
--> 6d50e2521fdd
Step 3/7 : RUN apt-get update
--> Using cache
--> 43c11ef00f95
Step 4/7 : RUN apt-get install -y gcc
--> Using cache
--> ca18aba18efb
Step 5/7 : RUN apt-get install -y g++
--> Using cache
--> b5de65c9486a
Step 6/7 : RUN g++ 2_1.cpp -static
--> Running in 4d0048250373
Removing intermediate container 4d0048250373
--> 2e93f29e4001
Step 7/7 : CMD ./a.out
--> Running in 482d3b24f1a5
Removing intermediate container 482d3b24f1a5
--> d0cbbb21af37
Successfully built d0cbbb21af37
Successfully tagged test:latest
```

Рисунок 2.3 – Сборка образа на основе Докерфайла

```
ruslan718@ruslan718-VirtualBox:~$ docker run -it test
4179
```

Рисунок 2.4 – Запуск образа

Заключение

В ходе выполнения данной работы было произведено ознакомление с основными способами передачи параметров подпрограмм, особенностями передачи управления между модулями; изучение комбинированных программ, в которых модули Ассемблера вызываются из модулей, написанных на высокоуровневых языках программирования.

Был создан docker-контейнер, с предустановленными компонентами компиляции и редактирования кода, необходимыми для написания программ на языке C++. В данной среде были написана и протестирована необходимая по заданию программа.

Приложение А

(обязательно)

Листинг кода на C++

```
#include <iostream>
#include <ctime>
#include <random>

using namespace std;

int main()
{
    srand(time(0));
    const int size = 10;
    int arr[size];
    int arr2[size];
    int sum;

    for (int i = 0; i < size; i++)
    {
        arr[i] = rand() % 51;
        arr2[i] = rand() % 51;
    }
    for (int i = 0; i < size; i++)
    {
        sum += arr[i] * arr2[i];
    }

    cout << sum << endl;
    return 0;
}
```

(обязательно)

```

mov ecx, arr[esi] //ложим в регистр ecx значение массива
imul ecx, arr2[esi] // перемножаем ecx на значение из второго массива, то есть ecx = ecx * arr2[esi]
add sum, ecx ; // результат умножения лежит в ecx, его ложим в sum
add esi, 4

    CheckEnd:
add dl, 1 ; // Увеличиваем количество пройденных строк
cmp dl, BYTE PTR size_asm ; // Сравнение, есть ли ещё строки
jl Start ; // Если есть, опять идем к функции Start

.att_syntax
)"
);

cout << sum << endl;
return 0;
}

```

Приложение В

(обязательно)

Dockerfile


```
FROM ubuntu
COPY 2_1.cpp .
RUN apt-get update
RUN apt-get install -y gcc
RUN apt-get install -y g++
RUN g++ 2_1.cpp -static
CMD ./a.out
```