Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
(НИУ ВШЭ)

Московский институт электроники и математики им. А.Н. Тихонова

ОТЧЕТ
О ПРАКТИЧЕСКОЙ РАБОТЕ № 2.1
по дисциплине «Архитектура компьютера и системное программирование»
ЗНАКОМСТВО С АССЕМБЛЕРОМ И РАЗНОЯЗЫКОВЫМИ ПРОГРАММАМИ

Студент гр. БИБ232
Аникиев Алексей Александрович

Руководитель
Абдуллаев Ариф Биннатович

Дата: 28.11.2025

Москва 2025

Содержание

1. Задание на практическую работу

Цели настоящей практической работы заключаются в следующем:

1. Познакомиться со структурой программы на языке Ассемблер, разновидностями и назначением сегментов, способами организации простых и сложных типов данных, изучить форматы и правила работы с транслятором MASM / FASM /TASM, компоновщиком и отладчиком, познакомиться с средствами создания программ на Ассемблере для ОС Linux.
2. Исследование способов связывания разноязыковых модулей.

Задание настоящей практической работы заключаются в следующем:

1. Основы программирования на ассемблере:
   1.1. Повторно ознакомиться с теоретическим материалом.
   1.2. С помощью языка программирования Ассемблер, используя любой синтаксис (Intel или AT&T) на свой выбор, необходимо написать программу, решающую задачу из варианта. При этом:
      1.2.1. Решением задачи является запись в специально объявленную для этого переменную верного результата при любых допустимых входных данных.
      1.2.2. Программа должна корректно завершаться, не вызывая аварийный останов.
      1.2.3. Объявлять входные данные необходимо самостоятельно в секции данных. Например, если в задаче указано «дан массив из 10 байт», в секции данных необходимо самостоятельно объявить такой массив, заполнить его байтами на своё усмотрение и использовать в качестве входного. В ходе защиты может возникнуть необходимость изменения таких данных.
      1.2.4. Для выполнения повторяющихся однотипных действий необходимо использовать соответствующие средства языка. Копирование и вставка одного и того же кода нужное количество раз с незначительными изменениями без необходимости является ошибкой.
      1.2.5. Программа должна выдавать корректный ответ для всех видов входных данных, описанных в задании. Например, если в задании написано «дан массив

из 10 байт», то входными данными может быть массив из любых 10 значений от 0 до 255. Учитывайте возможные переполнения!

1.2.6. При выполнении (в т.ч. дополнительных заданий) нельзя вызывать готовые ассемблерные подпрограммы из каких-либо библиотек и модулей.

2. Комбинирование разноязыковых модулей:

2.1. Повторно ознакомиться с теоретическим материалом

2.2. С помощью языков программирования С и Ассемблер, используя любой синтаксис (Intel или AT&T) на свой выбор, необходимо написать программу, решающую задачу из варианта:

2.2.1. Входные данные для задач, основная часть которых подразумевает обработку чисел, а не их создание, должны генерироваться автоматически с помощью ГПСЧ языка Си. Сгенерированные исходные данные должны быть выведены в консоль.

2.2.2. При случайной генерации чисел пользователю должна предоставляться возможность ограничить диапазон значений этих чисел.

2.2.3. Решением задачи является вывод в консоль верного результата при любых допустимых входных данных.

2.2.4. Программа должна корректно завершаться, не вызывая аварийный останов.

2.2.5. В программе на языке С должен быть реализован только следующий функционал: выделение памяти под исходные данные (и их инициализация в случаях, когда она должна быть произведена с помощью ГПСЧ, или ввод данных), вызов подпрограммы, выполняющей основные вычисления, вывод результата вычислений в консоль. Все основные вычисления должны быть оформлены в виде ассемблерного модуля (не ассемблерной вставки в код на Си, а именно отдельного файла на языке Assembler).

2.2.6. Для выполнения повторяющихся однотипных действий необходимо использовать соответствующие средства языка. Копирование и вставка одного и того же кода нужное количество раз с незначительными изменениями без необходимости является ошибкой.

2.2.7. Все программы должны работать с любыми размерами входных данных. Размер входных данных (даже если сами данные генерируются

случайно)должен задаваться пользователем посредством ввода с командной строки. Запрещается ограничивать максимальный размер входных данных.

3. Дополнительные задания:
   3.1. Вывести результирующий элемент данных в консоль (в окно «вывод» в IDE), используя только команды «чистого» ассемблера (не вызывая подпрограммы, написанные не самостоятельно, наподобие printf). Даёт два дополнительных балла.
   3.2. Выполнить все поставленные задачи, написав две программы с использованием обоих синтаксисов (Intel и AT&T). Даёт один дополнительный балл.
   3.3. Скомбинировать разноязыковые модули для ОС Linux и ОС Windows, используя любой компилятор, кроме GCC/G++ (а также иных средств, входящих в пакет GCC), и синтаксис Intel. При выполнении данного задания необходимо подробно пошагово отразить все этапы в отчёте. Даёт два дополнительных балла.

2. Ход работы

2.1. Основы программирования на ассемблере

Мой порядковый номер в группе: 2.

Задание: Дано двойное слово. Найти количество нулей в его двоичной записи. Результат поместить в отдельный элемент данных.

2.1.1. Алгоритм

1. Сдвинуть число влево командой shl на 1.
2. В зависимости от CF (будет содержать вытесненный бит числа) обновлять счетчик.
3. Вывести результат в виде шестнадцатеричного числа в консоль

2.1.2. Процесс разработки

1. Разрабаотка на ОС на базе Linux, для Linux
2. Написание исходного кода (приложен к ответу на практическую работу) синтаксисами Intel и AT&T.
3. Сборка для Intel синтаксиса средствами nasm,и ld для AT&T средством as и ld.

```
$ nasm -f elf64 pr1_1.asm
$ ld -o pr1_1 pr1_1.o
```

Рисунок 2.1.2.1 – Сборка для синтаксиса Intel

```
$ as -o pr1_1_atnt.o pr1_1_atnt.s
$ ld pr1_1_atnt.o -o pr1_1_atnt
```

Рисунок 2.1.2.2 – Сборка для синтаксиса AT&T

4. Написание Makefile для автоматизации сборки (приложен к ответу на практическую работу). Будет включать две цели: intel и atnt

2.1.3. Тестирование

Рассмотрим два числа, их бинарные записи (32 бита) и число нулей в них.

Таблица 1. Числа для проверки

Hex	Bin	Число нулей
0x1337beef	0b00010011001101111011111011101111	11
0xdeadcafe	0b11011110101011011100101011111110	10

Проверим работоспособность программы на них:

```
[oleg@yolki 1]$ ./pr1_1
0x0000000B
```

Рисунок 2.1.3.1 – Проверка ппрограммы для числа 0x1337beef

```
[oleg@yolki 1]$ ./pr1_1
0x0000000A
```

Рисунок 3.1.3.2 – Проверка ппрограммы для числа 0xdeadcafe

Программа отрабатывает верно.

2.2. Комбинирование разноязыковых модулей

Мой порядковый номер в группе: 2.

Задание: Напишите программу, в которой создается и заполняется натуральными числами (1, 2, 3 и далее), поочерёдно возведёнными во вторую или третью степень, двумерный массив. Заполнение начинается с левого верхнего элемента слева направо, сверху вниз (то есть заполняется сначала первая строка, затем вторая, и так далее). Поочерёдное возведение во вторую или третью степень означает, что первый элемент возводится во вторую степень, второй – в третью, третий – снова во вторую, четвёртый – снова в третью и т.д.

2.2.1. Алгоритм

1. Часть C:

1.1. Запросить у пользователя максимальный размер массива

1.2. Сгенерировать случайные размеры массива

1.3. Выделить память массива в сооствествии со сгенерированными числами

1.4. Передать управление функции, реализованной на ассемблере

1.5. После выполнения функции освободить выделенную память

2. Часть ASM:

2.1. Выделить часть стека

2.2. Итерироваться последовательно по строкам двумерного массива

2.3. Инициализировтать ячейки массива необходимыми числами

2.4. Вывод в консоль массива

2.2.2. Процесс разработки

1. Разрабаотка на ОС на базе Linux, для Linux
2. Написание исходного кода (приложен к ответу на практическую работу) синтаксисами Intel и AT&T.
3. Сборка для Intel синтаксиса средствами nasm и clang для AT&T средствами clang.

```
$ nasm -f elf64 asm.asm
$ clang -o task asm.o main.c
```

Рисунок 2.2.2.1 - Сборка для синтаксиса Intel

```
$ clang main.c asm_atnt.s
```

Рисунок 4.2.2.2 – Сборка для
синтаксиса AT&T

4. Написание Makefile для автоматизации сборки (приложен к ответу на практическую работу). Будет включать две цели: intel и atnt

2.2.3. Тестирование

Проверим работу программы:

```
[oleg@yolki 2]$ ./a.out
Input rand_max for array sizes > 10
0x0000000000000001 0x0000000000000008 0x0000000000000009 0x0000000000000040 0x0000000000000019 0x00000000000000D8 0x0000000000000031 0x0000000000000200 0x0000000000000051 0x00000000000003E8
0x0000000000000079 0x00000000000006C0 0x00000000000000A9 0x0000000000000AB8 0x00000000000000E1 0x0000000000001000 0x0000000000000121 0x0000000000016C8 0x0000000000000169 0x0000000000001F40
0x00000000000001B9 0x0000000000002998 0x0000000000000211 0x0000000000003600 0x0000000000000271 0x00000000000044A8 0x00000000000002D9 0x000000000000055C0 0x0000000000000349 0x0000000000006978
0x00000000000003C1 0x0000000000008000 0x0000000000000441 0x0000000000009988 0x000000000000004C9 0x000000000000B640 0x0000000000000559 0x000000000000D658 0x00000000000005F1 0x0000000000000FA00
0x0000000000000691 0x0000000000012168 0x0000000000000739 0x000000000014CC0 0x00000000000007E9 0x0000000000017C38 0x00000000000008A1 0x000000000001B000 0x0000000000000961 0x000000000001E848
```

Рисунок 5 – Запуск программы с ограничениями размера массива 10

Программа работает корректно

3. Выводы о проделанной работе

В ходе практической работы были изучены основы программирования на ассемблере и реализованы алгоритмы обработки данных с использованием низкоуровневых инструкций. Также было освоено взаимодействие модулей на C и ассемблере, что позволило понять принципы комбинирования разноязыковых программ.