Лабораторное занятие 3. Получить ПЕРВУЮ ПРОГРАММУ (изначально используя язык записи алгоритмов ассемблер) ДЛЯ 64-РАЗРЯДНОЙ ЭВМ, которая **может заставить ЭВМ (БЕЗ ИСПОЛЬЗОВАНИЯ ОБОЛОЧКИ DOSBOX) вывести на экран знакомое Вам стандартное окно: MessageBox, имеющее все стандартные «пимпочки», а в качестве сообщения в него выводится текст: «Hello, world», а после нажатия клавиши ввода это сообщение изменяется на «Hello, hell!».**

**Внимательно читаем постановку задачи, чтобы не «попасть»!!!**

**Н**а прошлом лабораторном занятии Вы должны были тщательно проштудировать Лекцию 5 из Лекций Зубовича и «Помыть папу»! А также нужно было выполнить лабораторную работу №1 «Вывести на экран изображения всех символов кодировочной таблицы, используемой на Вашем компьютере», проштудировав Лекцию 23 всё из тех же Лекций Зубовича. Надеюсь, что у большинства из Вас это получилось. Упор на первых двух занятиях был сделан на том, чтобы понять, что при решении любой задачи по «принуждению» компьютера ВЫПОЛНИТЬ строго определенную последовательность действий, необходимо осуществить следующую последовательность действий:

1. Набор исходного текста на языке записи алгоритмов (например, на языке assembler).
2. Перевод исходного текста с помощью программного средства под названием ТРАНСЛЯТОР на машинный язык (в текст, представляющий собой совокупность команд и данных, записанных на языке машины – в двоичной системе счисления) с получением так называемого «объектного модуля» - файла типа obj.
3. Так называемая ЛИНКОВКА, в рамках которой осуществляется НАСТРОЙКА адресов областей памяти, куда будет загружаться ПРОГРАММА, и настройка-привязка-линковка-подключение так называемых внешних функций, с получением, в зависимости от используемых параметров-ключей команды LINK, так называемого исполнимого модуля (файла) типа exe или типа com.
4. После получения такого файла осуществляется его ЗАГРУЗКА ПОД УПРАВЛЕНИЕМ ОПЕРАЦИОННОЙ СИСТЕМЫ (но НЕ операционной оболочки) В ОПЕРАТИВНУЮ ПАМЯТЬ с получением ПРОГРАММЫ, хранящейся в ней, и с последующей передачей управления этой программе по указанной ТОЧКЕ ВХОДА (передача управления также осуществляется ОПЕРАЦИОННОЙ СИСТЕМОЙ).

Цитирую окончание Лекции 5 из Лекций Зубовича, чтобы продемонстрировать описанный выше процесс превращения мысли в голове в то, что называется ПРОГРАММОЙ.

«Для того, чтобы у Вас и у меня хоть что-то осталось в голове после нее, нарисуем графически и очень кратко то, чем мы занимались на протяжении 50 (пятидесяти) рисунков, нарисовав еще один – пятьдесят первый (Рис. 5.51).

**MicroSoft Word Tasm.exe**

Исходный текст, сохраненный в файлле Mrak.txt ( в папке на диске)

То, что придумали, записалли на листочке

**Редактор Это диск СЖ Транслятор**

То, что есть Преобразование

на листочке исходного модуля в

Mrak.obj

Mrak.Lst, если все нормалльно

переносим в объектный модуль с

Персоналлку получением листинга

Это Вы и Мы!

Думаем, как получить

Программу Если есть ошибки, то снова нужно

Редактировать исходный текст

**Если при линковке обнаружены ошибки, Теперь из *объектного* нужно получить**

**то снова идем в редактор *исполнимый***

**Tlink.exe**

**Mrak.exe если все нормально**

**Линковщик**

## Результат

**Вместе с «Лодырем» и «Мимом» Mrak.exe**

**DPMILOAD.exe DPMIMEM.dll Запуск программы**

**Пусть это дисплей, на**

**Котором нужно искать результат**

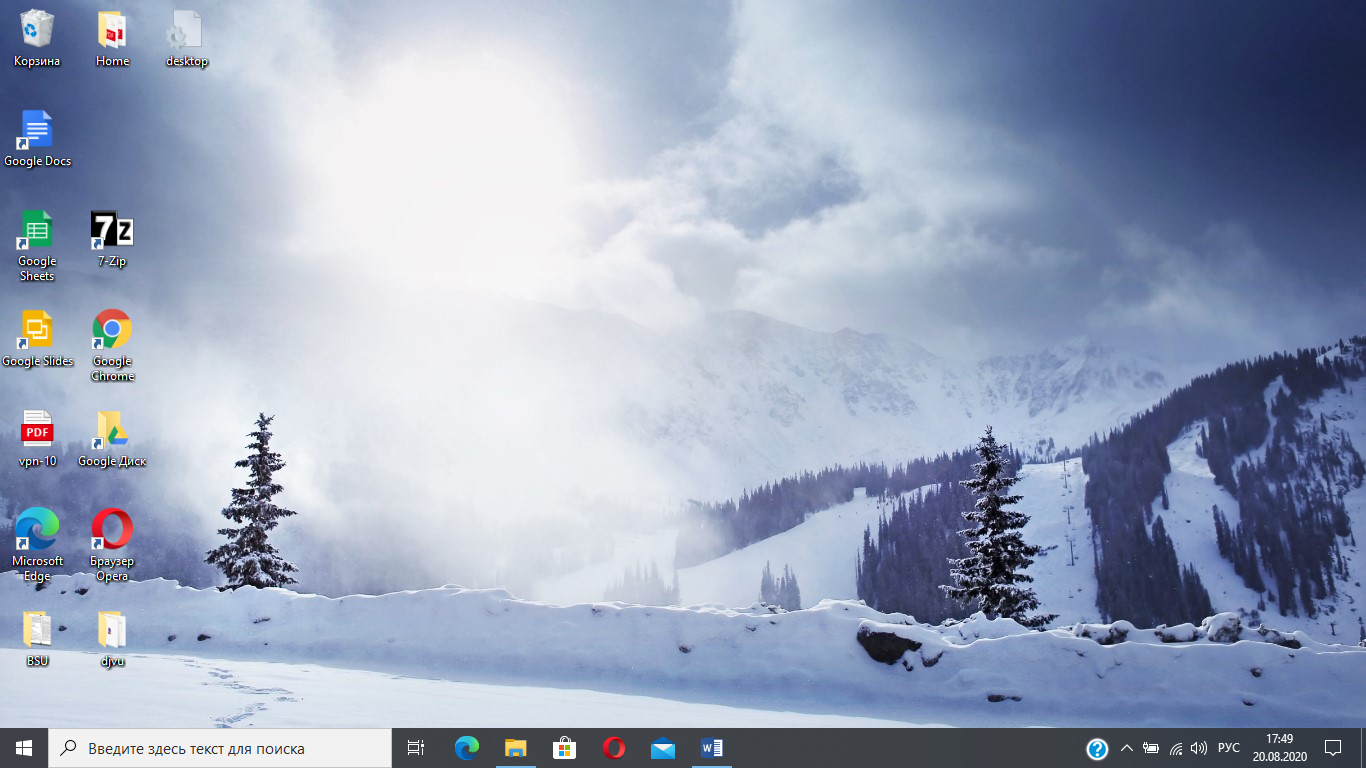
**Если результат не хорош ! Эта песня хороша, начинай сначала!** *Если то, что Вы видите*

*В качестве результата, совпадает с тем, что Вы желали увидеть – Радуйтесь, такое бывает редко*

**Рис. 5.51. Процесс трансляции, линкования и выполнения программ на языке ассемблер**

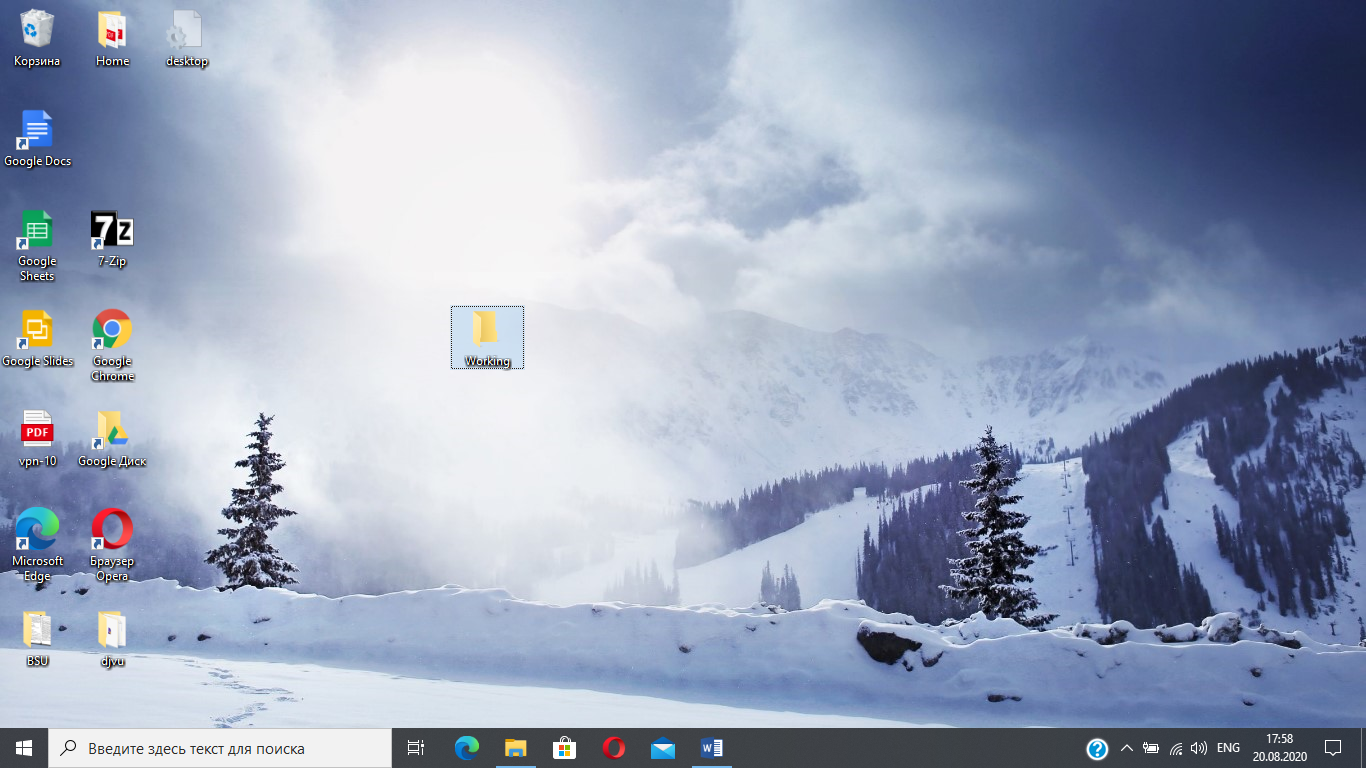
**Вы, наверное, уже не удивитесь, если я скажу, что для «продвинутых» 64-разрядных машин и соответствующих им операционным оболочкам типа Windows 10 (дэсяць), процесс набора исходного текста на языке assembler, линкования с получением соответствующего исполнимого модуля-файла с последующей его загрузкой под управлением операционной системы НИЧЕМ НЕ ОТЛИЧАЮТСЯ ОТ ПРЕДСТАВЛЕННОГО ВЫШЕ. Отличия, заключаются ТОЛЬКО В ТОМ, что в связи с «прогрессом», будут использованы другие трансляторы, линковщики, файлы. Давайте, например, продемонстрируем сий факт на примере решения следующей задачи. Необходимо получить программу, которая может заставить ЭВМ вывести на экран знакомое Вам стандартное окно: MessageBox, имеющее все стандартные «пимпочки», а в качестве сообщения в него выводится текст: «Hello, world», а после нажатия клавиши ввода это сообщение изменяется на «Hello, hell!». Начали…**

***Вот так выглядит рабочий стол моего компьютера:***



**Если вспомнить предыдущую лекцию и описание процесса редактирования исходного текста на языке записи алгоритмов assembler, трансляции с получением соответствующего текста в виде команд на машинном языке, «линчевание» объектного модуля (соответствующего текста в виде команд на машинном языке) с получением так называемого исполнимого модуля и последующей загрузки этого исполнительного модуля в оперативную память под управлением операционной системы (НЕ ОБОЛОЧКИ, а СИСТЕМЫ!) с получением программы и передачей ей управления….**

**Так что сначала пишем на листочке исходный текст, сохраняя этот текст в каком-либо текстовом файле в какой-либо «рабочей папке», например, «working». На Рис. 6.1. Вы можете видеть появление этой папки на моём рабочем столе:**



**Рис. 6.1. Появление папки «Working» на рабочем столе.**

**Теперь наберем исходный текст на языке записи алгоритмов, который после трансляции, линкования, получения исполнимого модуля с последующей загрузкой его в оперативную память ПРЕВРАТИТСЯ В ПРОГРАММУ – последовательность машинных команд или инструкций, которая после ДЕКОДИРОВАНИЯ ИХ самой машиной, заставит выполнить эту машину требуемую последовательность действий: сформировать на экране компьютера в рамках операционной оболочки windows”10 так называемый messageBox с сообщением «Hello, world», а после нажатия на «пимпочку»-«батон» ОК, поменять на «Hello, Hell!».**

**Для набора текста воспользуемся простым как палка программным средством под названием «Блокнот»**

**; Sample x64 Assembly Program**

**extrn MessageBoxA: PROC**

**.data**

**caption db '64-bit hello!', 0**

**message db 'Hello World!', 0**

**.code**

**Start proc**

**sub rsp,28h ; shadow space, aligns stack**

**mov rcx, 0 ; hWnd = HWND\_DESKTOP**

**lea rdx, message ; LPCSTR lpText**

**lea r8, caption ; LPCSTR lpCaption**

**mov r9d, 0 ; uType = MB\_OK**

**call MessageBoxA ; call MessageBox API function**

**; mov ecx, eax ; uExitCode = MessageBox(...)**

**mov message+6,'h'**

**mov message+7,'e'**

**mov message+8,'l'**

**mov message+9,'l'**

**mov message+10,'!'**

**mov caption+11,'!'**

**; add rsp,28h ; shadow space, aligns stack**

**mov rcx, 0 ; hWnd = HWND\_DESKTOP**

**lea rdx, message ; LPCSTR lpText**

**lea r8, caption ; LPCSTR lpCaption**

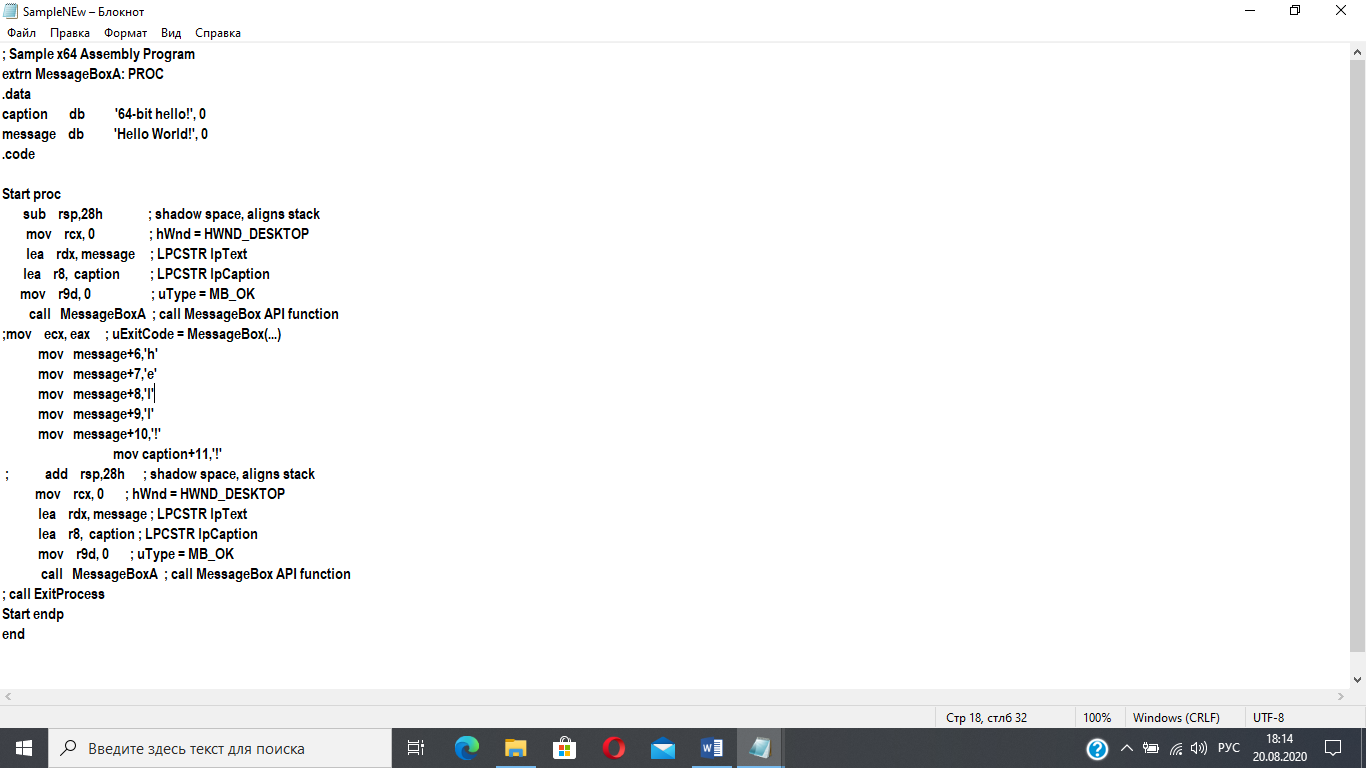
**mov r9d, 0 ; uType = MB\_OK**

**call MessageBoxA ; call MessageBox API function**

**; call ExitProcess**

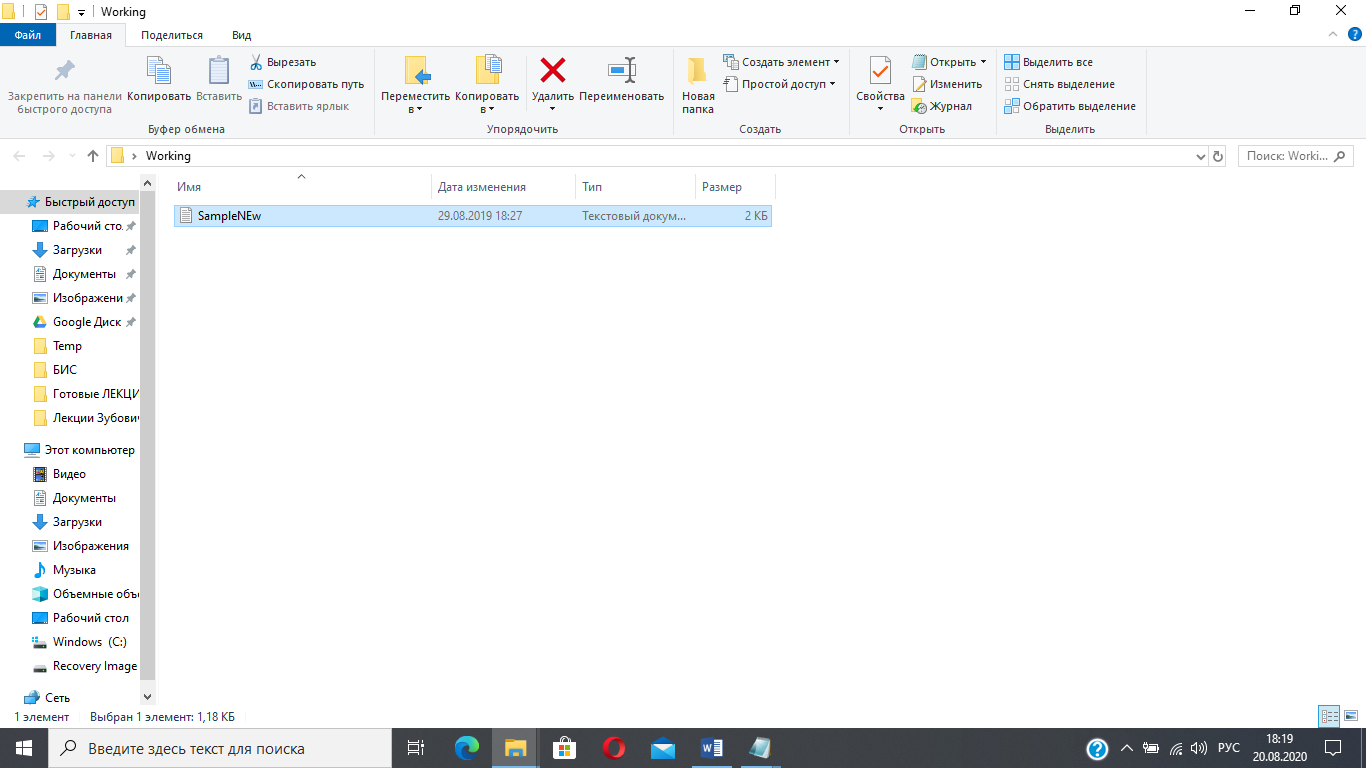
**Start endp**

**End**



**Рис. 6.2. Исходный текст на языке записи алгоритмов assembler.**

**Понятно, что исходный текст называется «SampleNew» и этот исходный текст необходимо сохранить в папке Working. Делаем.**



**Рис. 6.3. Файл SampleNew в папке Working.**

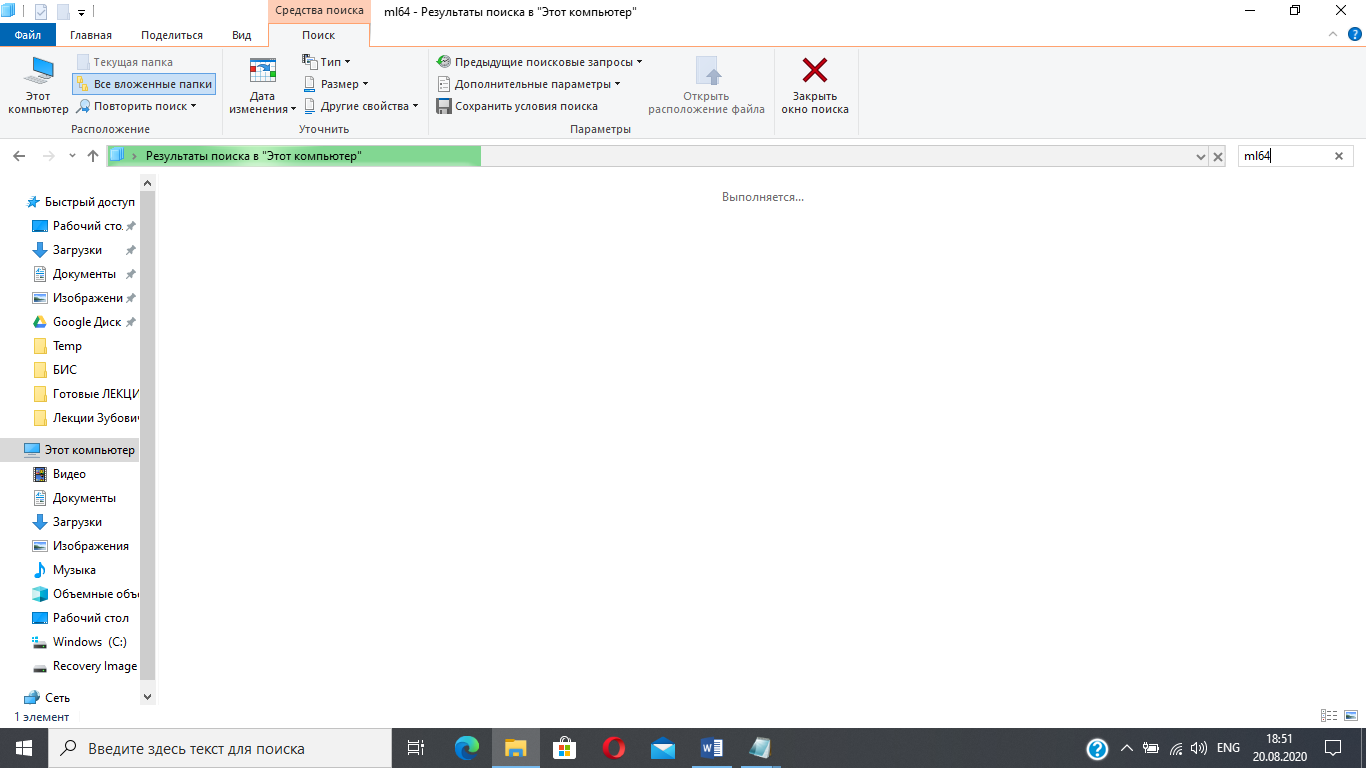
**ВНИМАНИЕ! В предыдущем тексте СДЕЛАНО КАК МИНИМУМ ДВЕ ОБМАНКИ!!! Необходимо их найти и в процессе выполнения работы на этом занятии сразу же после обнаружения выслать мне на почту сообщение – какие две обманки были сделаны….**

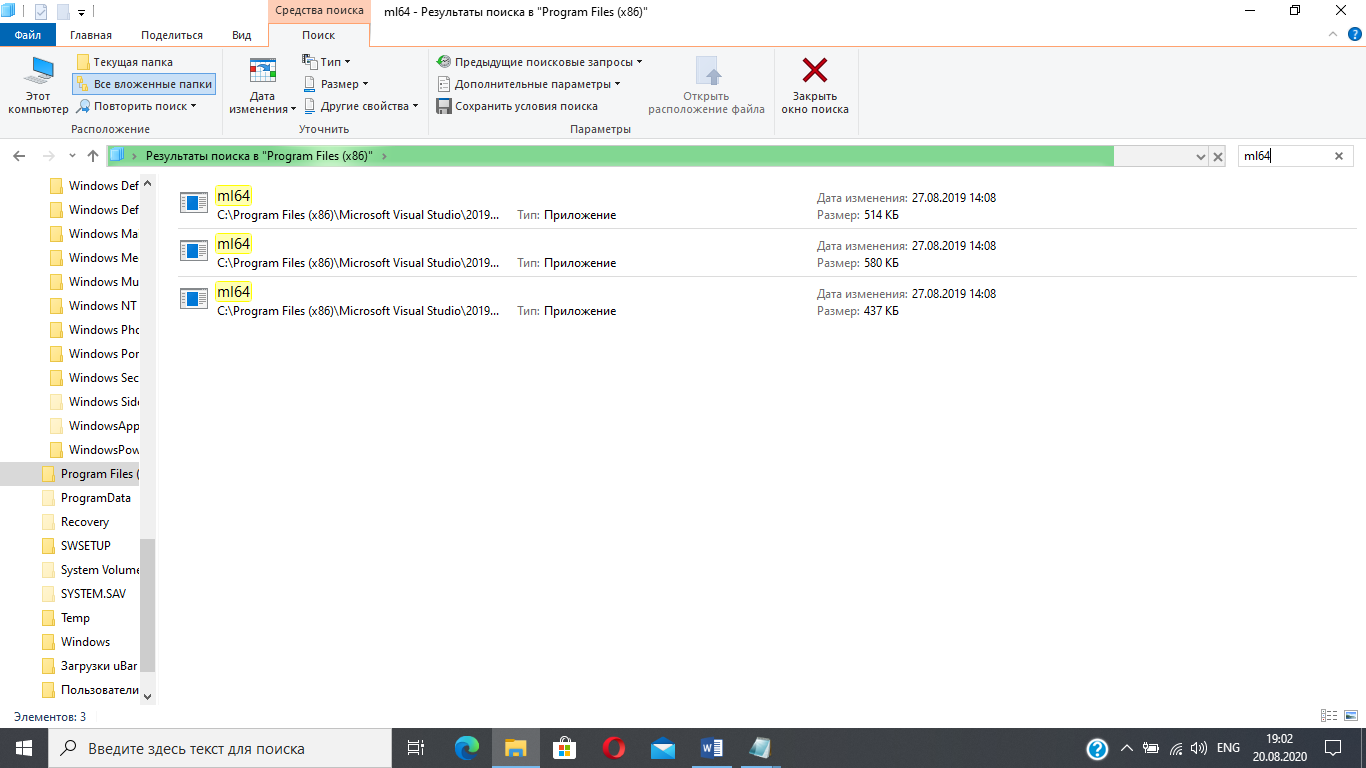
**Теперича нашей задачей является НАХОЖДЕНИЕ ХОТЬ КАКОГО-НИКАКОГО «захудаленького» исполнимого модуля, который мог бы ассоциироваться с тем, что называется программным средством «транслятор» для преобразования исходных текстов, написанных на языке записи алгоритмов assembler, в текст на машинном языке, который называется «объектным модулем», непонятно, правда, почему. Причем при поиске нужно иметь ввиду, что машина у нас 64-разрядная, а выполнение КОМАНД ОПЕРАЦИОННОЙ СИСТЕМЫ (для диалога с «конечным» пользователем, человеком, сидящем за компьютером) будет проходить в рамках ОБОЛОЧКИ Windows”10.**

**Возникает вопрос у каждого представителя подрастающего поколения: «А где искать?». Ребята! ТОЛЬКО НЕ НА ПОМОЙКЕ!!!!! ИСКАТЬ НА ПОМОЙКЕ ЗАПРЕЩЕНО!!!! Разрешено только использовать данные с сайтов ОРИГИНАЛЬНЫХ РАЗРАБОТЧИКОВ ТЕХ ИЛИ ИНЫХ ПРОГРАММНЫХ СРЕДСТВ!!!!! Для программных средств, которые чаще всего используются в мире, таковым разработчиком является – ни за что не догадаетесь, что это за организация – MicroSoft, как ни странно. На портале Microsoft.com есть раздел под названием docs.microsoft.com, которым разрешается Вам пользоваться, но на языке оригинала. И если кратко ознакомиться с содержанием этих документов, то можно найти, что в рамках того, что продвигается Microsoft под «шильдой» Visual Studio 2019, есть большое количество данных о том, что и как нужно делать, чтобы в рамках этого «проекта» Visual Studio 2019 можно было бы использовать аппарат получения программ с применением языка записи алгоритмов, команды которого максимально приближены к аппаратному уровню.**

**В частности, в этих документах можно найти, что в недрах того, что называется программный продукт MicroSoft Visual Studio 2019 (в большом количестве папок, различных файлов со страшными именами и расширениями), спрятан такой исполнимый модуль с названием ml64, который как раз и представляет собой ТРАНСЛЯТОР, осуществляющий перевод исходных текстов, записанных на языке assembler, в текст на машинном языке.**

**А давайте попробуем «пашукаць» это самое ml64.**

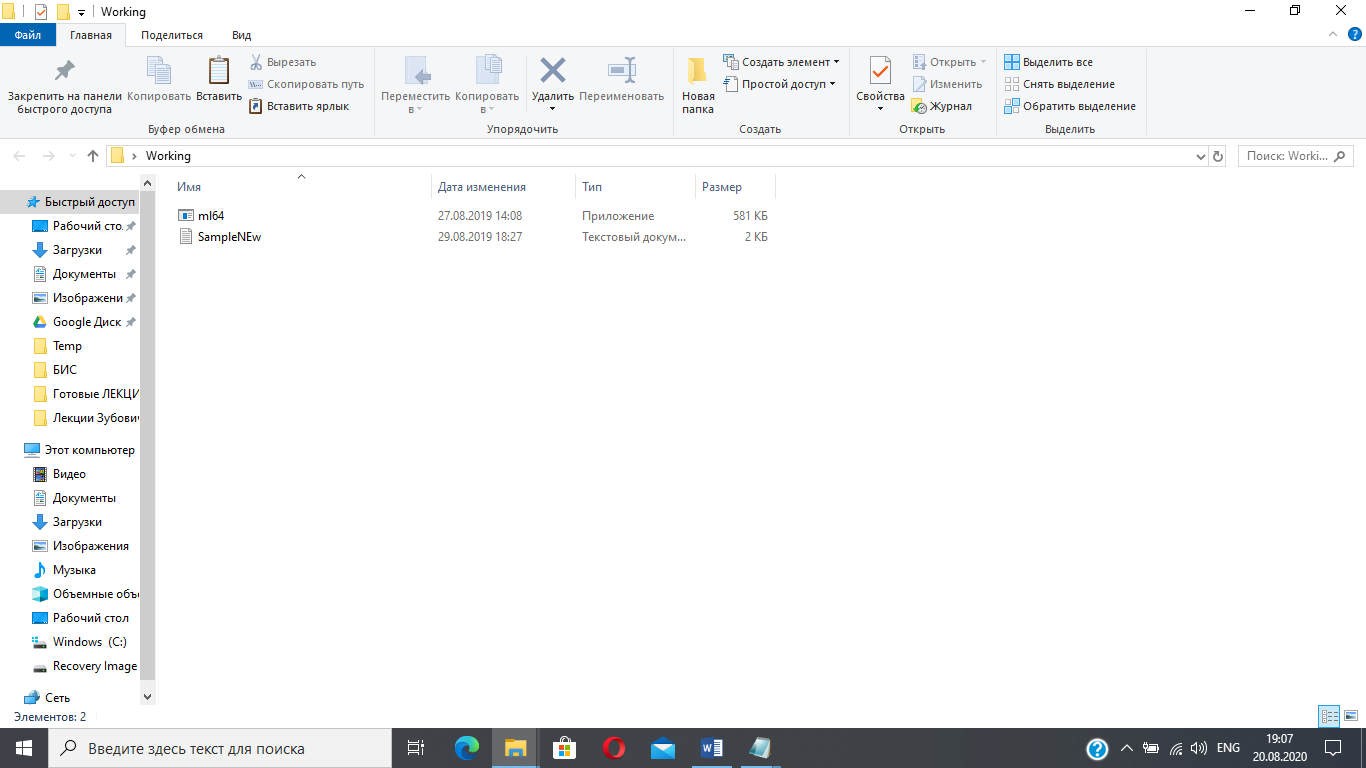




**И вот что найдено…. Очень интересно! Аж три файла с одним и тем же именем, но с различными размерами и написаны слова, которые Вам произносить ЗАПРЕЩАЕТСЯ!!!! Эти все файлы – ИСПОЛНИМЫЕ МОДУЛИ!!!!! Возвращаясь же к трём файлам с одинаковыми именами, так и хочется воскликнуть – «ДВОЕЧНИКИ»!!!! А точнее, «вытягиватели» денежек из доверчивых конечных пользователей. Ведь говорят, что чем больше «кода» (это слово тоже произносить запрещено), тем дороже «продукт»…. Или это я чего-то не догоняю?!**

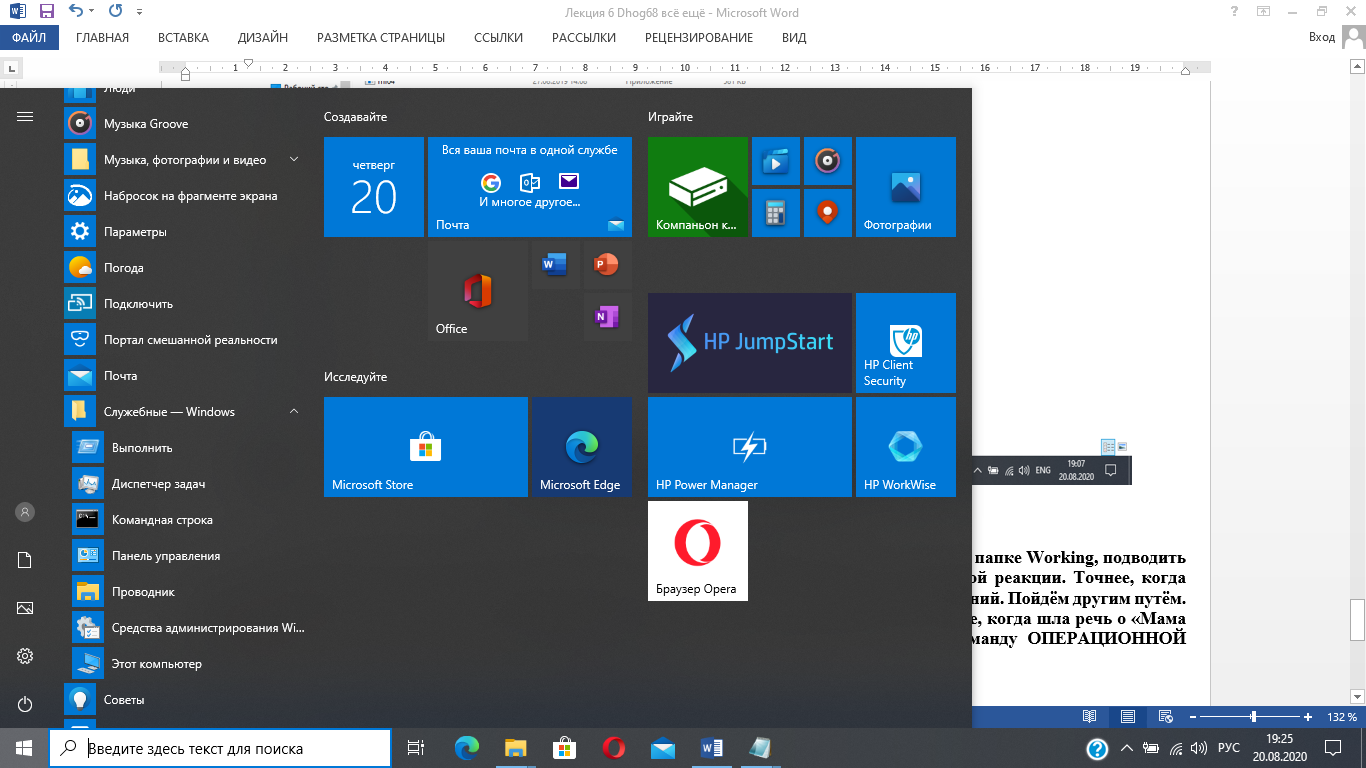
**И у Вас это есть!!! Обращаю также Ваше внимание, где этот ml сидит – ProgramFiles (x86)!!!!**

**Надеюсь, что у Вас получится скопировать файл из папки в папку и Вы получите в папке Working такую картинку:**



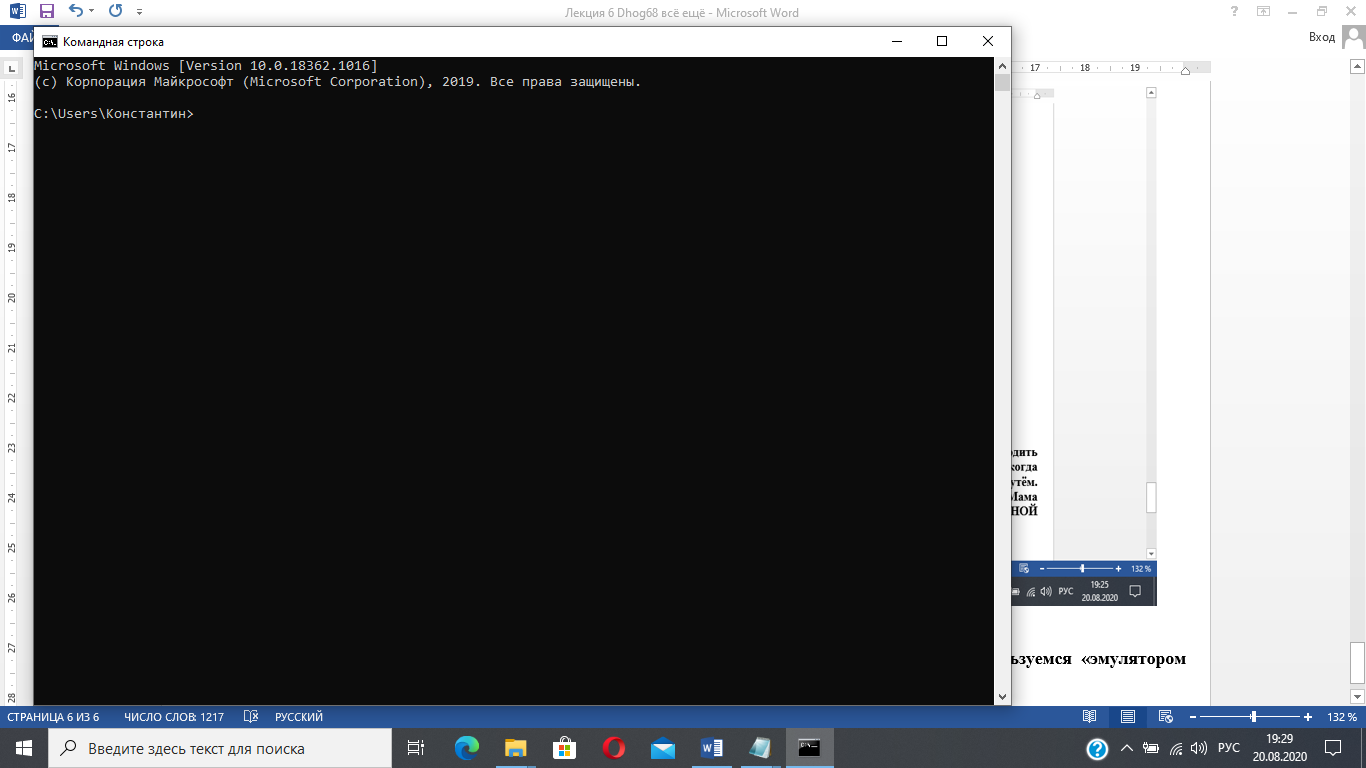
**Рис. 6.4. Результат поиска транслятора ml64 – исполнимого модуля.**

**Что дальше? Дальше, используя «возможности» оболочки Windows”10, будем, находясь в папке Working, подводить мышку к ML64 и жать на левую клавишу мышки, как учит «великий Win»! И что? Да никакой реакции. Точнее, когда нажимали первый раз, что-то мигнуло на экране, а дальше глухо как в танке – никаких телодвижений. Пойдём другим путём. А именно попытаемся сделать так, чтобы картинка была похожа на ту, которая была у нас ранее, когда шла речь о «Мама мыла ПаПу». Там мы могли ручками набирать команду не операционной оболочки, а команду ОПЕРАЦИОННОЙ СИСТЕМЫ без названия, используя окошко «Выполнить»….**



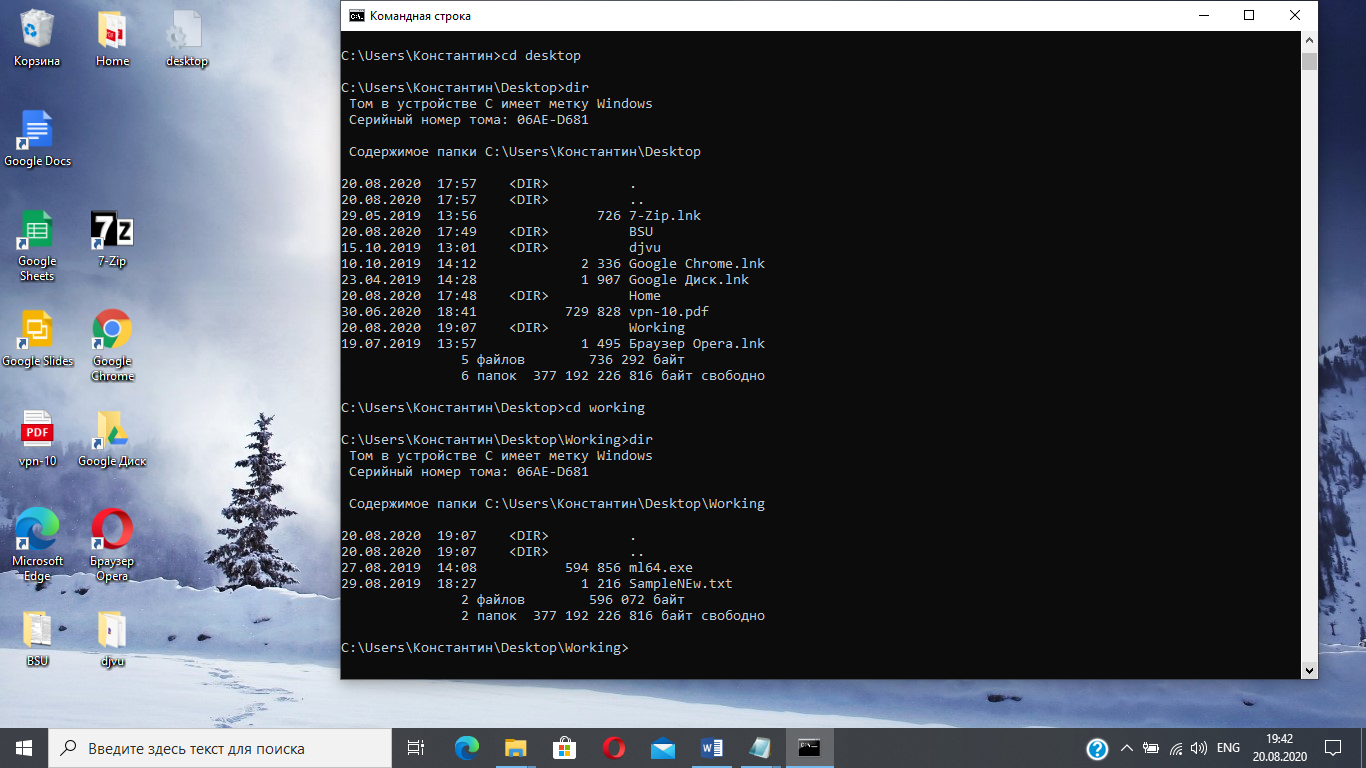
**Рис. 6.5. Где найти «Выполнить».**

**Но далее, для разнообразия, и чтобы считать себя более «продвинутыми» воспользуемся «эмулятором командной строки» в рамках того, что называется «Операционная оболочка Windows”10».**



**Рис. 6.6. «Обманка» командной строки.**

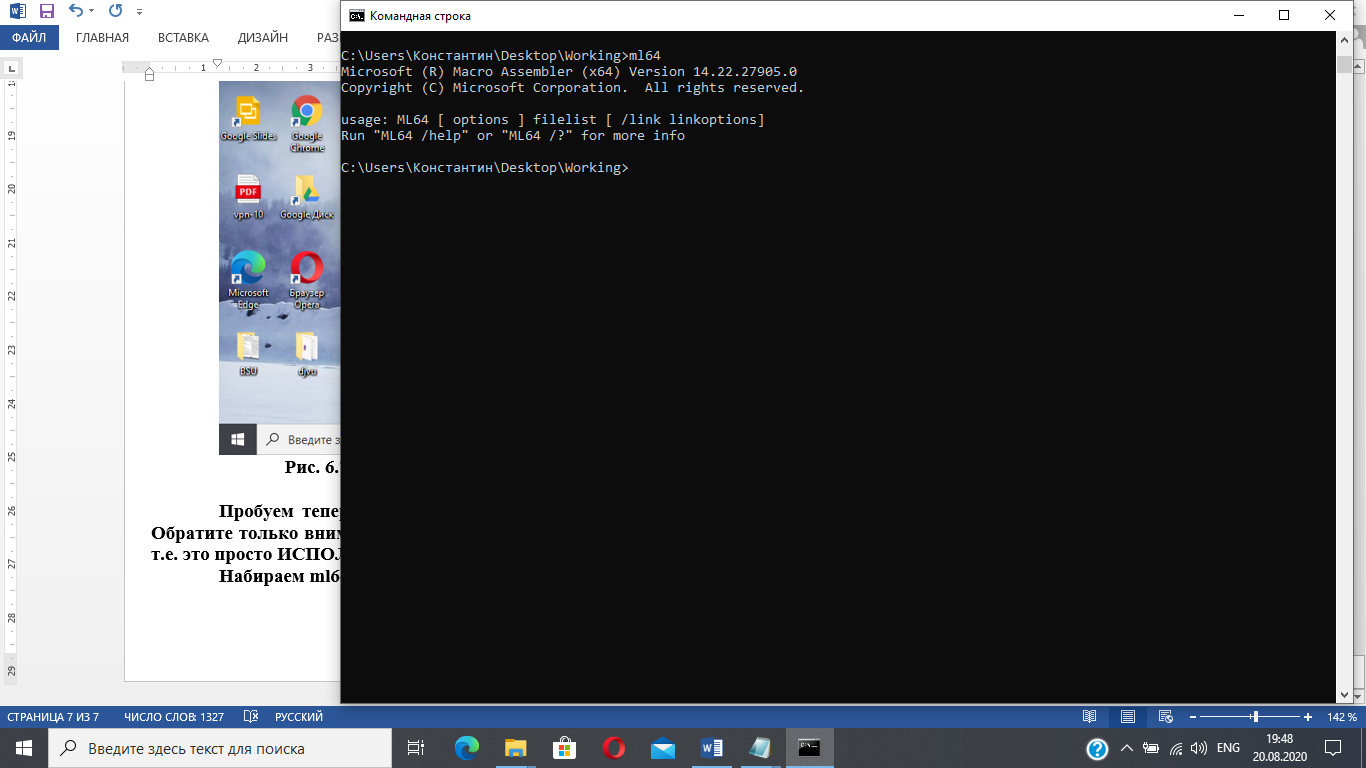
**Нам необходимо как-то сделать так, чтобы наша папка «Working» СТАЛА ТАК НАЗЫВАЕМОЙ АКТИВНОЙ, т.е. чтобы там, где написано: C:\Users\Константин>, было написано: …Working>. Где вместо трёх точек должен быть написан ПОЛНЫЙ ПУТЬ. Для перехода из папки в паку, для работы с папками и файлами используются команды операционной системы типа DIR, CD, MD… Попробуем…**



**Рис. 6.7. Как сделать папку «Working» активной. И её содержимое.**

**Пробуем теперь по аналогии с ранее рассмотренным примером с Tasm «запустить» транслятор ml64 (Рис. 6.8.). Обратите только внимание, что в командной строке этот ящик (точнее оболочка) пишет, что ML64 имеет расширение exe, т.е. это просто ИСПОЛНИМЫЙ МОДУЛЬ!!!!! Никакое НЕ ПРИЛОЖЕНИЕ!!!!**

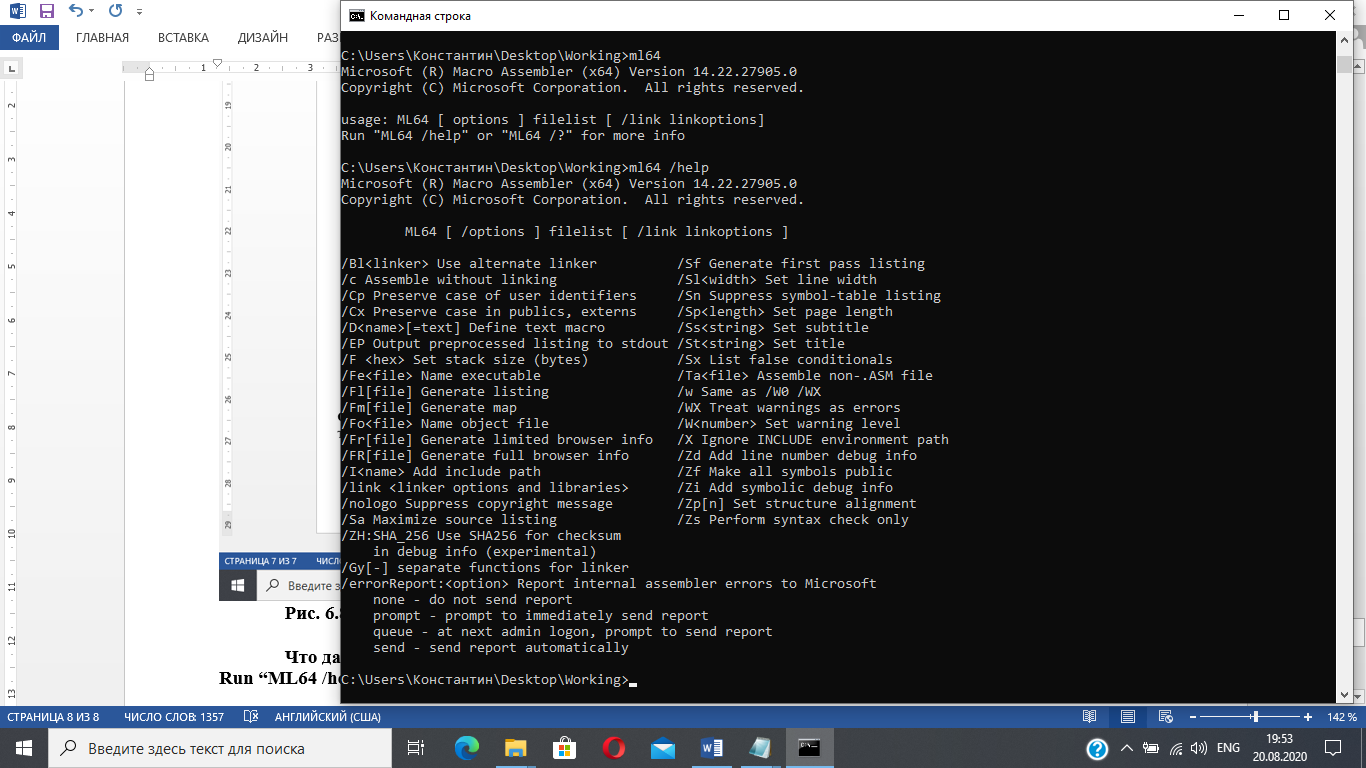
**Набираем ml64, получаем (ПОЛУЧИЛИ!!!!):**



**Рис. 6.8. Получили? Что получается, если просто набрать ml64.**

**Что дальше? Копаем головой отсюда и до умопомрачения, пытаясь рисовать. Следуем тому, что написано:**

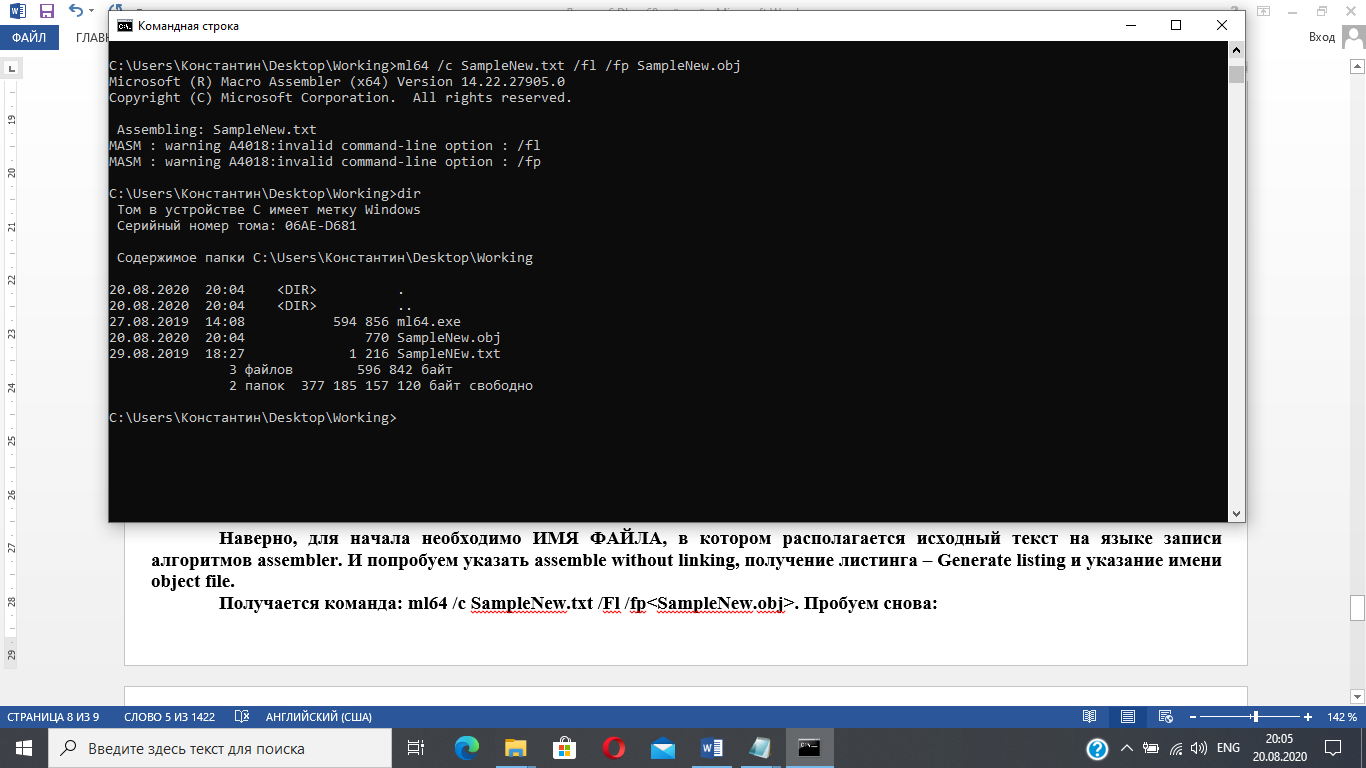
**Run “ML64 /help” for more info.**



**Рис. 6.9. Ужас! Без головы не разберёшься. Квадратные скобки указывают на необязательность опций, т.е. они могут присутствовать, а могут отсутствовать, но обязательно должен присутствовать filelist.**

**Наверно, для начала необходимо ИМЯ ФАЙЛА, в котором располагается исходный текст на языке записи алгоритмов assembler. И попробуем указать assemble without linking, получение листинга – Generate listing и указание имени object file.**

**Получается команда: ml64 /c SampleNew.txt /Fl /fp<SampleNew.obj>. Пробуем снова:**

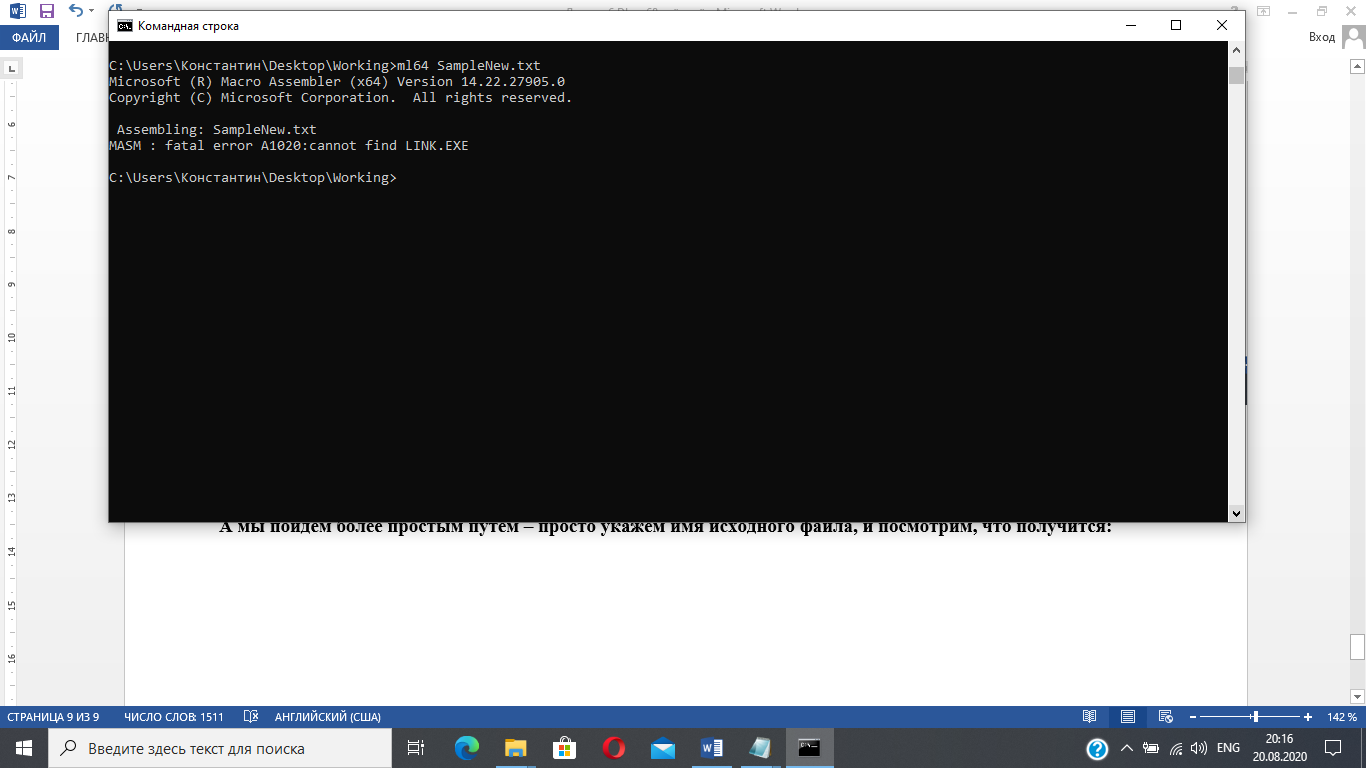


**Рис. 6.10. Первый блин полукомом. Программное средство транслятор ML64 обругало нас по поводу того, что invalid command-line option: /fl /fp. Но тем не менее, трансляция прошла успешно и получен файл SampleNew.obj. Похоже, что не так указана последовательность опций и файлов – ml64 подсказывал, что нужно сначала описать опции, а потом указать соответствующий список файлов. Хотелось бы посмотреть на листинг, поэтому делаем «попытку №2. Точнее.**

**Упражнение ещё одно. Получить листинг, а КАК ТОЛЬКО ПОЛУЧИТЕ ТУТ ЖЕ прислать его мне на почту [Zubovich20242025@mail.ru](mailto:Zubovich20242025@mail.ru).**

**Это своего рода проверка того, насколько Вы уважаете дедушку в смысле чтения его текстов, и как Вы блюдёте основное правило разработчика программного обеспечения – АДЕКВАТНО СЛЕДОВАТЬ ПОСТАНОВКЕ ЗАДАЧИ!!!! НЕ ШЛИТЕ ОТВЕТ НА ЭТО УПРАЖНЕНИЕ НА ПОЧТУ [chivobuz2025@mail.ru](mailto:chivobuz2025@mail.ru) – ghbik`nt – gjkexbnt rexe vbyecjd!!!**

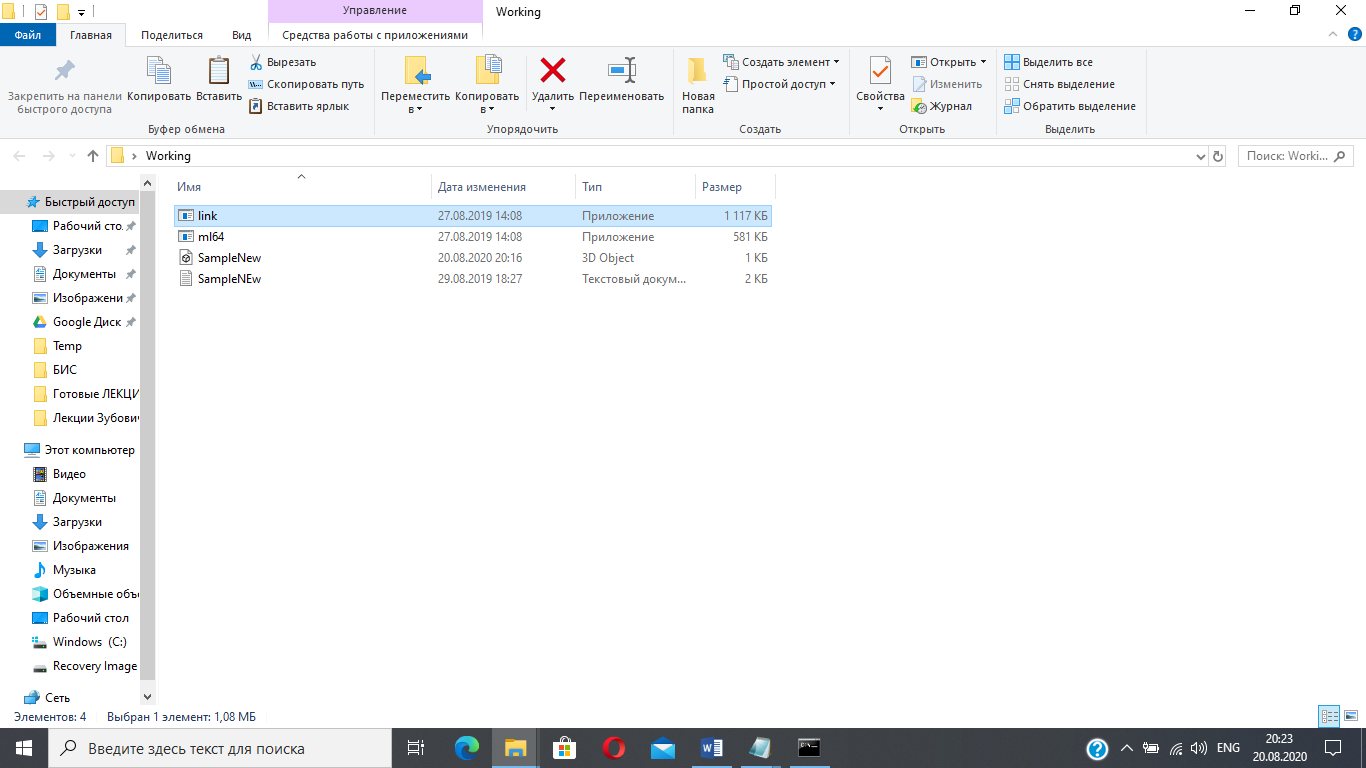
**А мы пойдём более простым путем – просто укажем имя исходного файла, и посмотрим, что получится:**



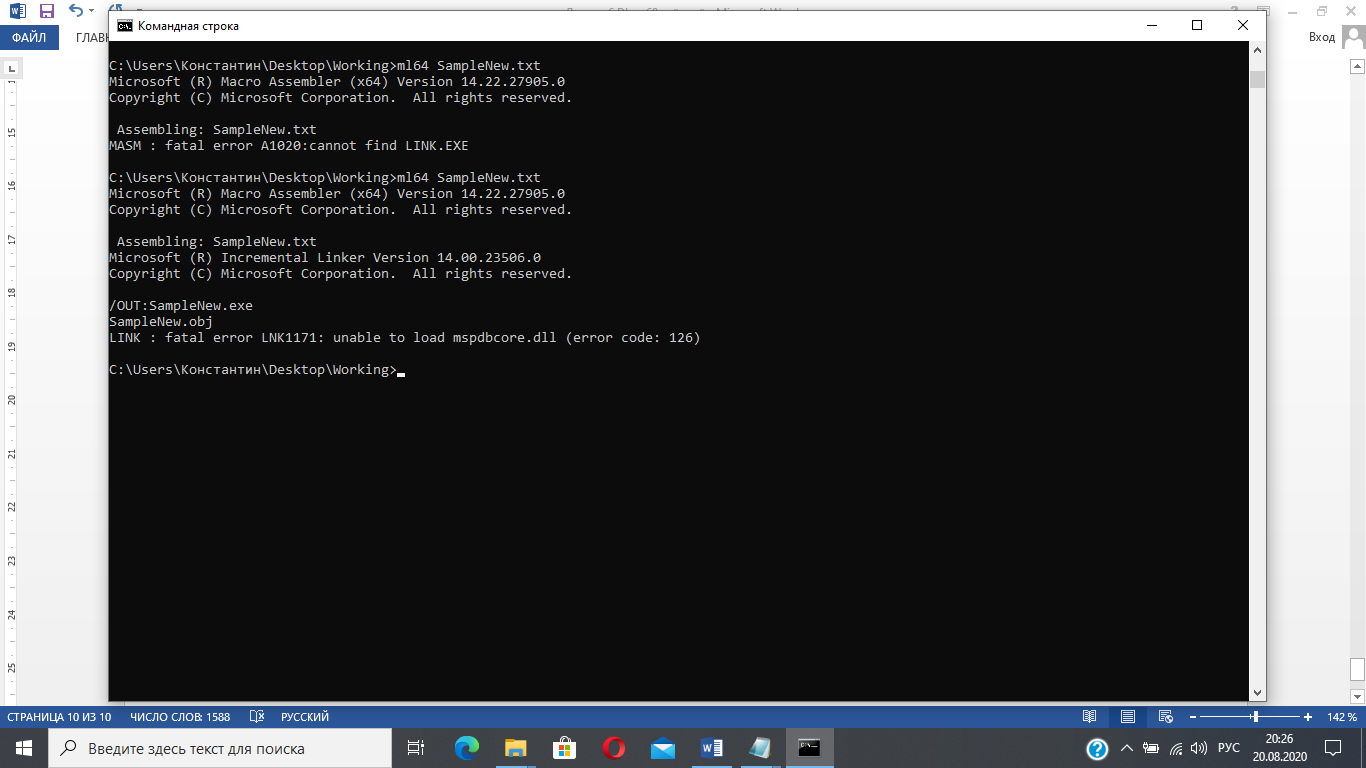
**Рис. 6.11. Оказывается, что если просто на вход ml64 подать имя файла, то будет сформирован 3D object файл, но при этом будет делаться попытка сделать ещё и Линковку! MASM: fatal error A1020:cannot find LINK.EXE.**



**Ха-ха-ха! Вот удивил этот транслятор! Не может найти LINK.EXE! Зато мы знаем, где искать!!! Найдя, размещаем этот файл в папке Working:**

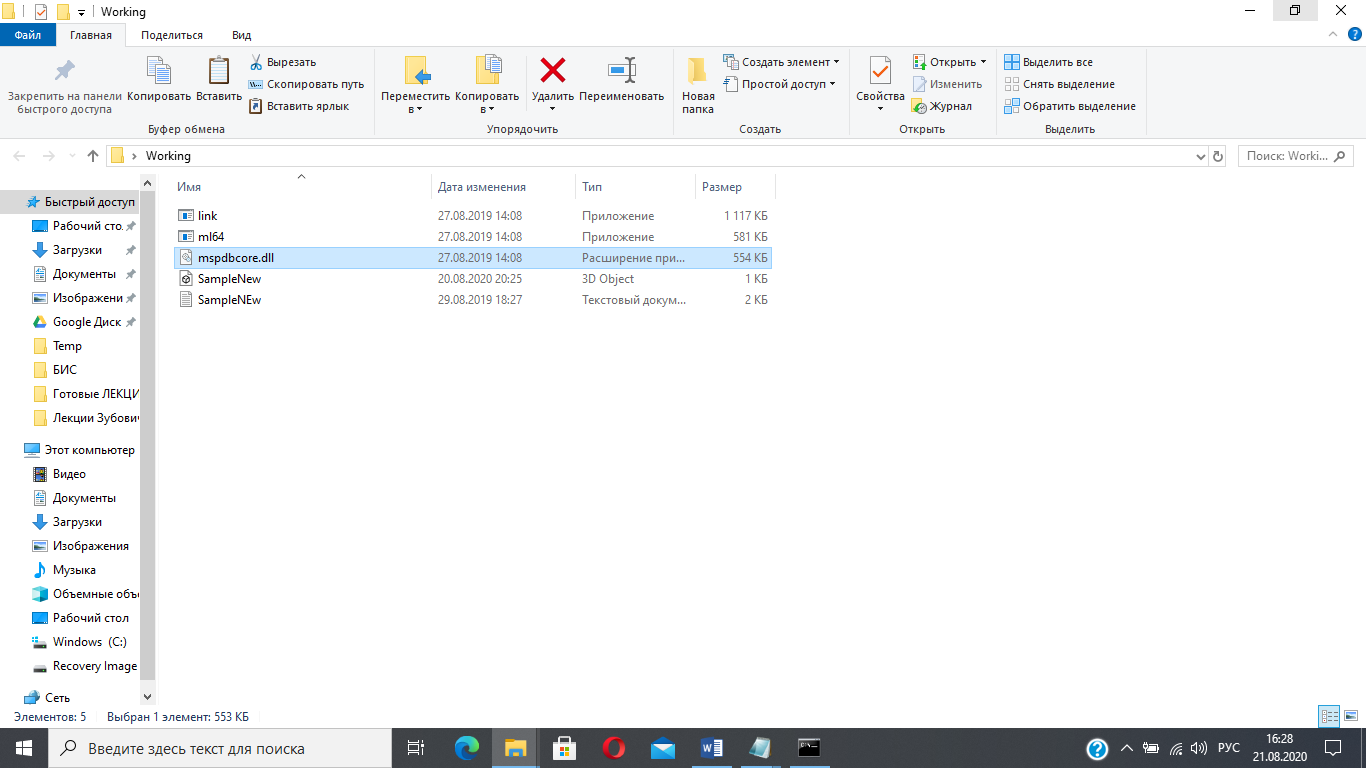


**Пробуем. Как Вы думаете, что будет дальше? Правильно! По аналогии с Лекцией 5 из Лекций Зубовича опять напишет, что чего-то не хватает в этом супе:**



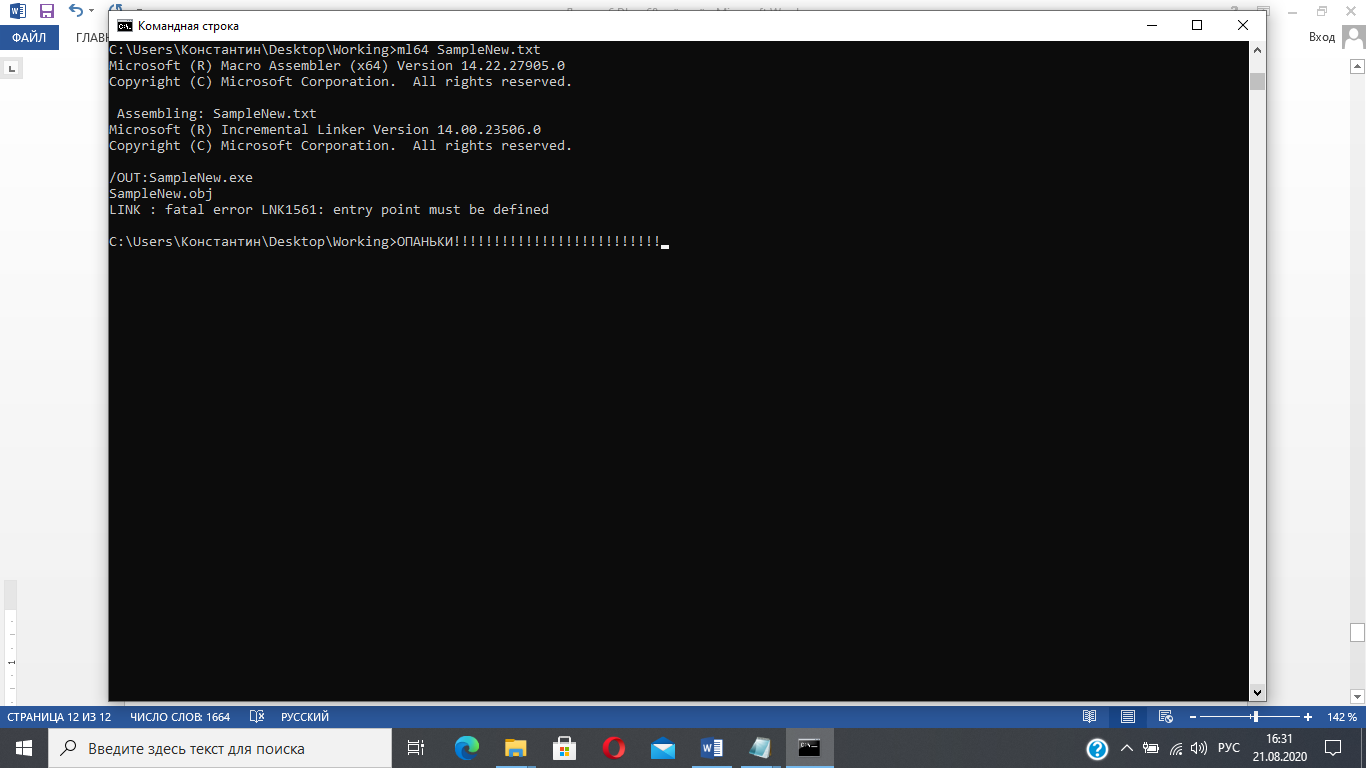
**Рис. 6.12. Вам это ничего не напоминает?**

**Мне напоминает. Только название такое «заковыристое» - mspdbcore – тяжело что-то придумать для русского аналога, как в случае с «лодырем». Но, не придумывая, ничего, и вспоминая, что «Только бледнолицый может наступить на грабли дважды», занимаемся поиском чего-то там “Microsoft programs data base core”. Громко, однако. Особенно про core – ядро?! Ищем. Находим. Размещаем в папке «Working» (Рис. 6.13.)**

**Р 6.13.**

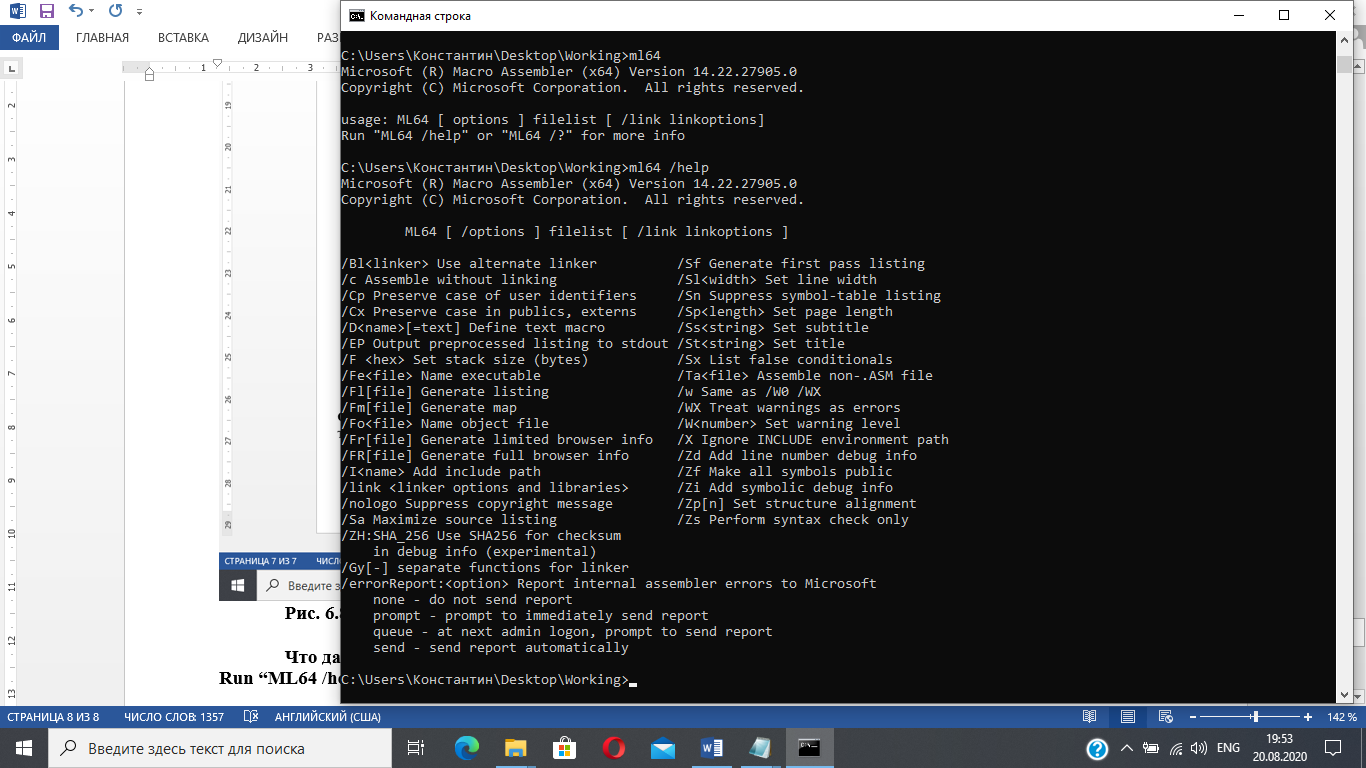
**Рис. 6.13. Какое-то dll, в имени которого имеется Core.**

**Снова пробуем:**

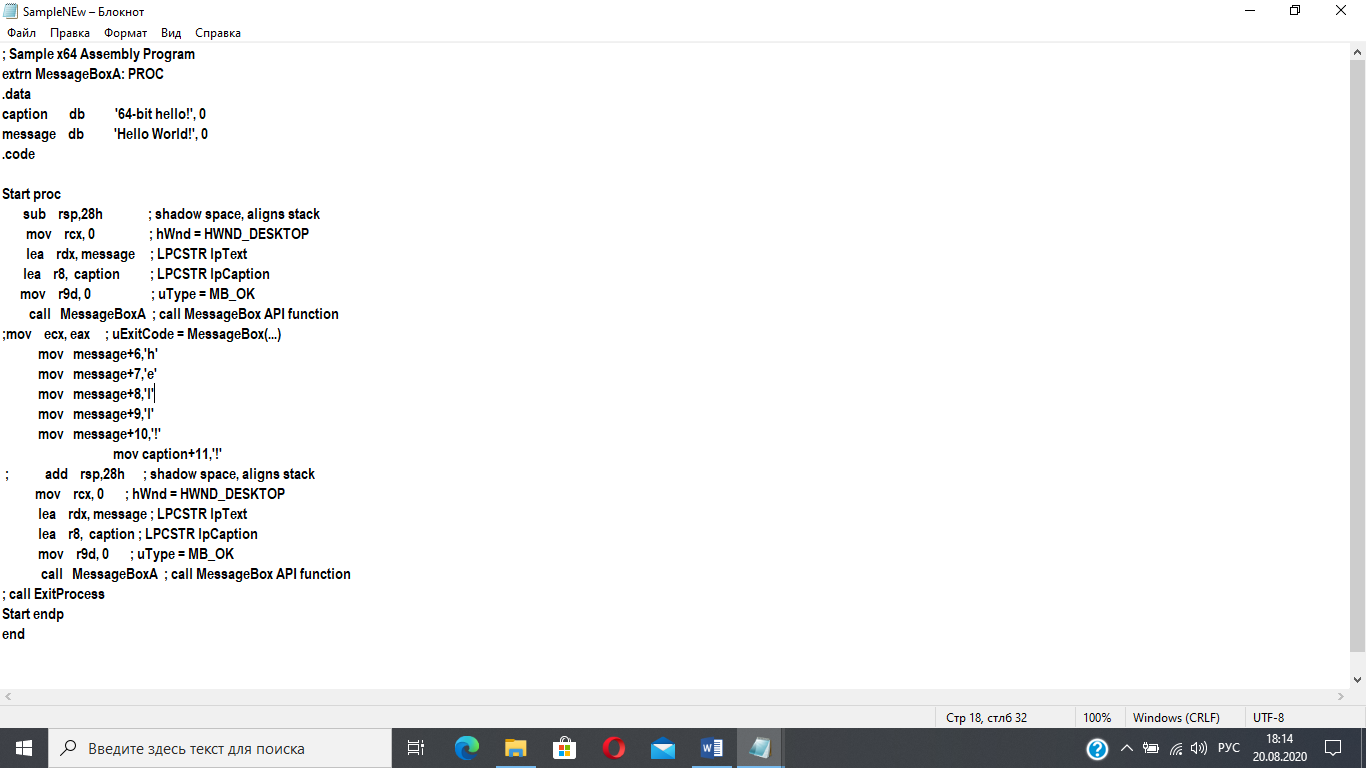


**«ОПАНЬКИ» - это я уже дописал. В прошлый раз (в Лекции 5 из Лекций Зубовича) такого не было, чтобы НУЖНО БЫЛО УКАЗЫВАТЬ КАКУЮ-ТО ТОЧКА ВХОДА ДА ЕЩЁ при вызове ТРАНСЛЯТОРА. Что-то мудрое. Приходится возвращаться к опциям ml64 и разбираться с «точкой входа», для чего понадобится исходный текст:**

**Это опции:**



**Это исходный текст:**

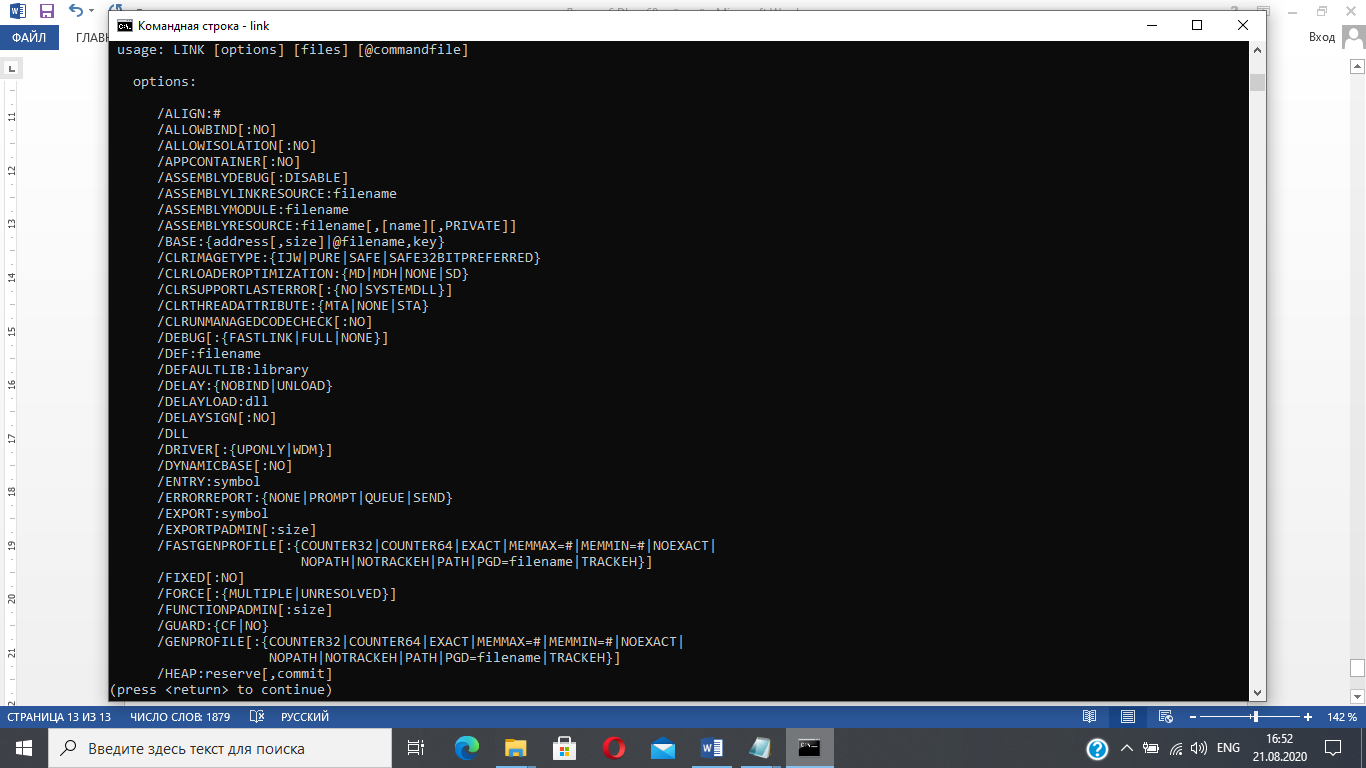


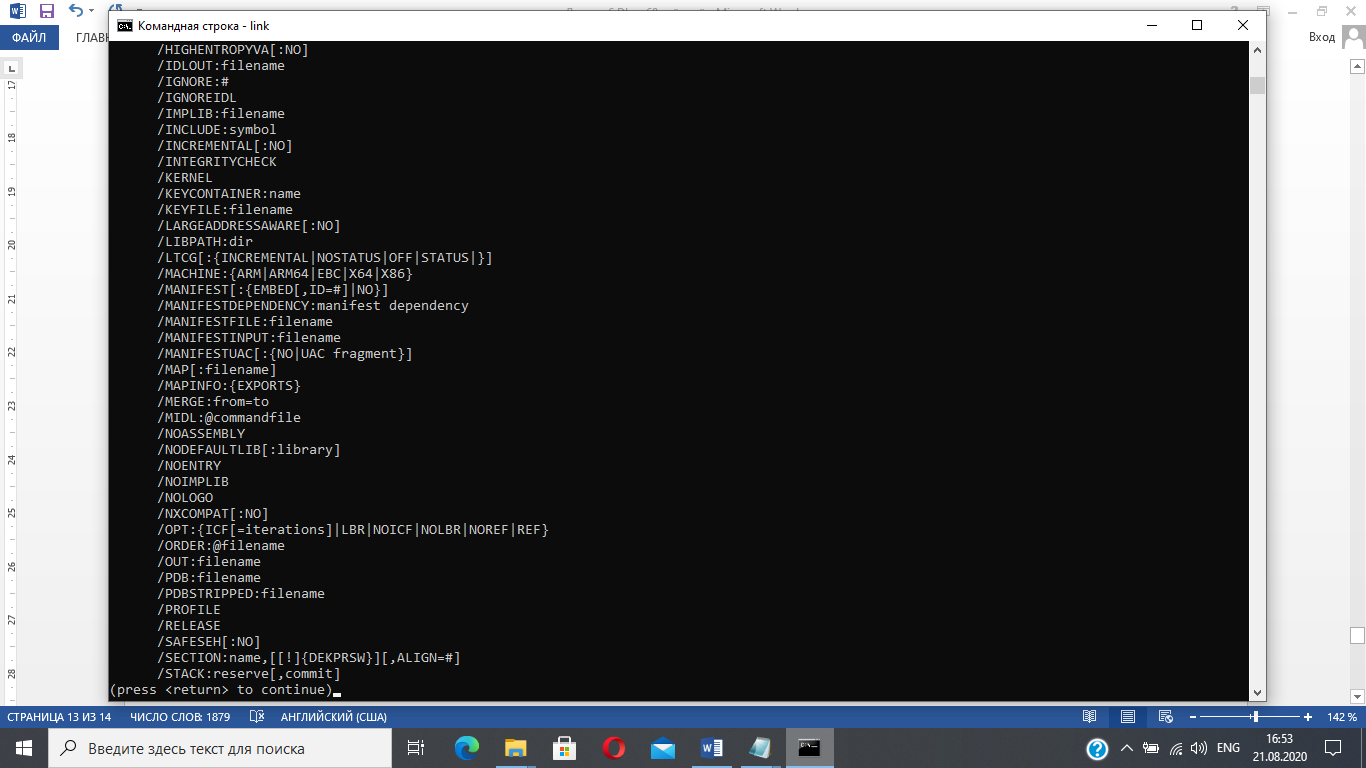
**\***

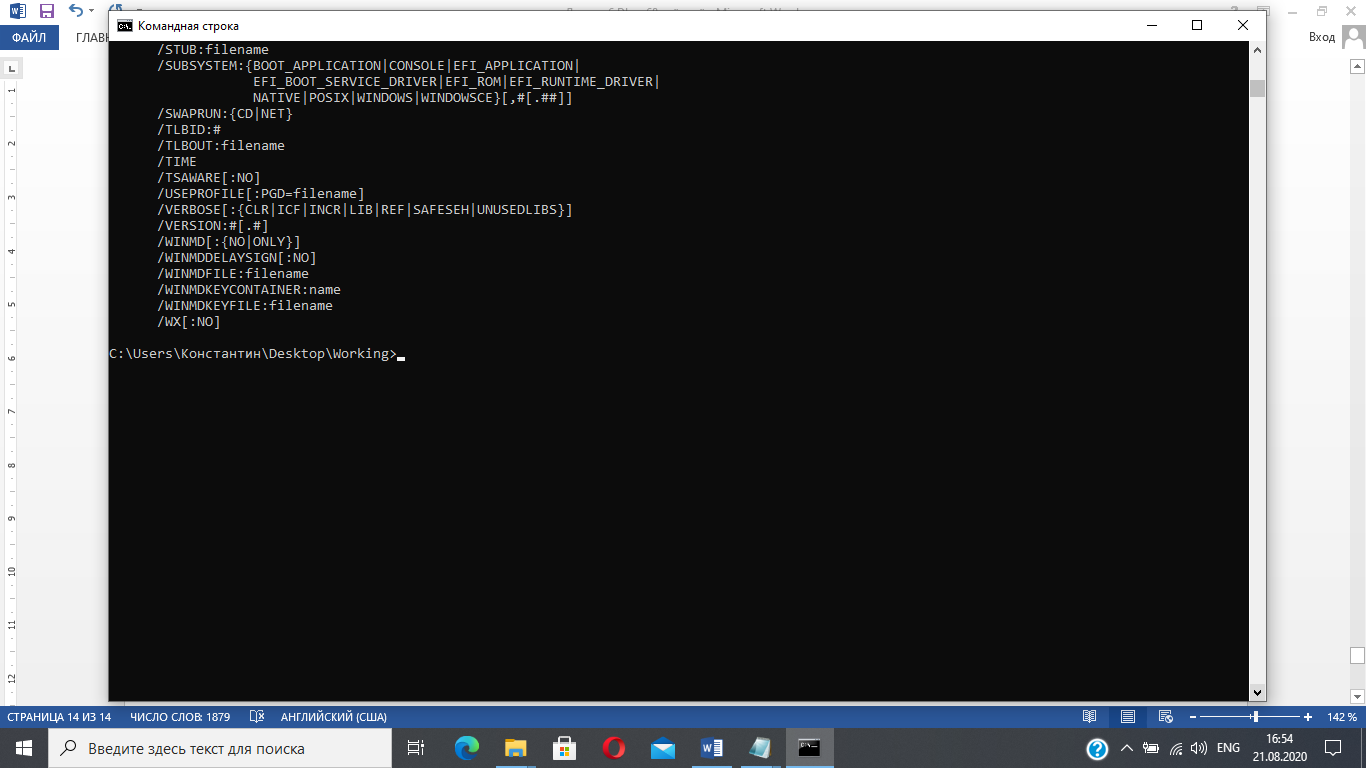
**Очень похоже, что ТОЧКОЙ ВХОДА следует называть АДРЕС ПЕРВОЙ МАШИННОЙ КОМАНДЫ, получаемый в результате трансляции и линковки исходного текста на машинный язык. Причем ранее говорилось, что всё, что используется в программе, располагается в оперативной памяти, и каждому понятию, использованному при записи на языке записи алгоритмов, транслятором ставится в соответствие АДРЕС – номер байта, начиная с которого будет располагаться данные или команды. В нашем случае можно видеть, что исходный текст предполагает, что ПЕРВОЙ ИСПОЛНИМОЙ КОМАНДОЙ (ТОЧКОЙ ВХОДА) должна быть команда, ассоциирующаяся с записью Start proc. То есть точкой входа в нашем случае является адрес команды, который будет сформирован транслятором с линковщиком при преобразовании исходного текста Start proc на машинный язык. Причем Start – это метка (символическое обозначение адреса, начиная с которого в оперативной памяти будет располагаться программа).**

**Смотрим, что у нас там с опциями транслятора ml64 и линковщика – каким образом указать, что точкой входа является то, что получается в результате преобразования метки Start. У ML64 есть опция /LINK <linker options and libraries>.**

**Пробуем определиться, какие опции есть у LINK. Путь нам известен:**

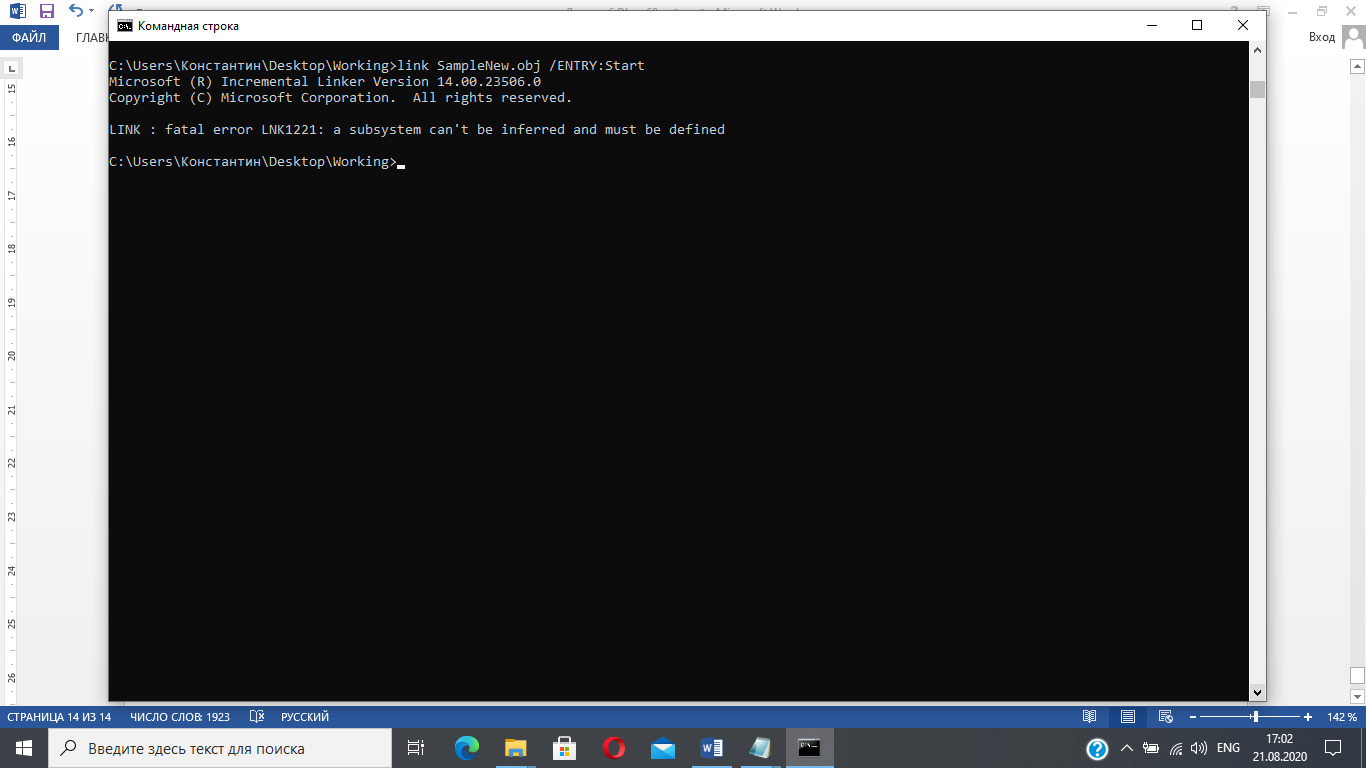






**Класс!!!! Сколько возможностей у Link! Какие неограниченные возможности, но нам от этого не легче. Нам просто нужно слинковать простенькую маленькую, а не какого-то монстра!**

**Похоже, что LINKу нужно как-то указать имя линкуемого файла (SampleNew.obj) и как-то указать точку входа – Start. И посмотреть, что будет…. Ох, и тяжелая это работа: из болота тянуть БЕГЕМОТА!**

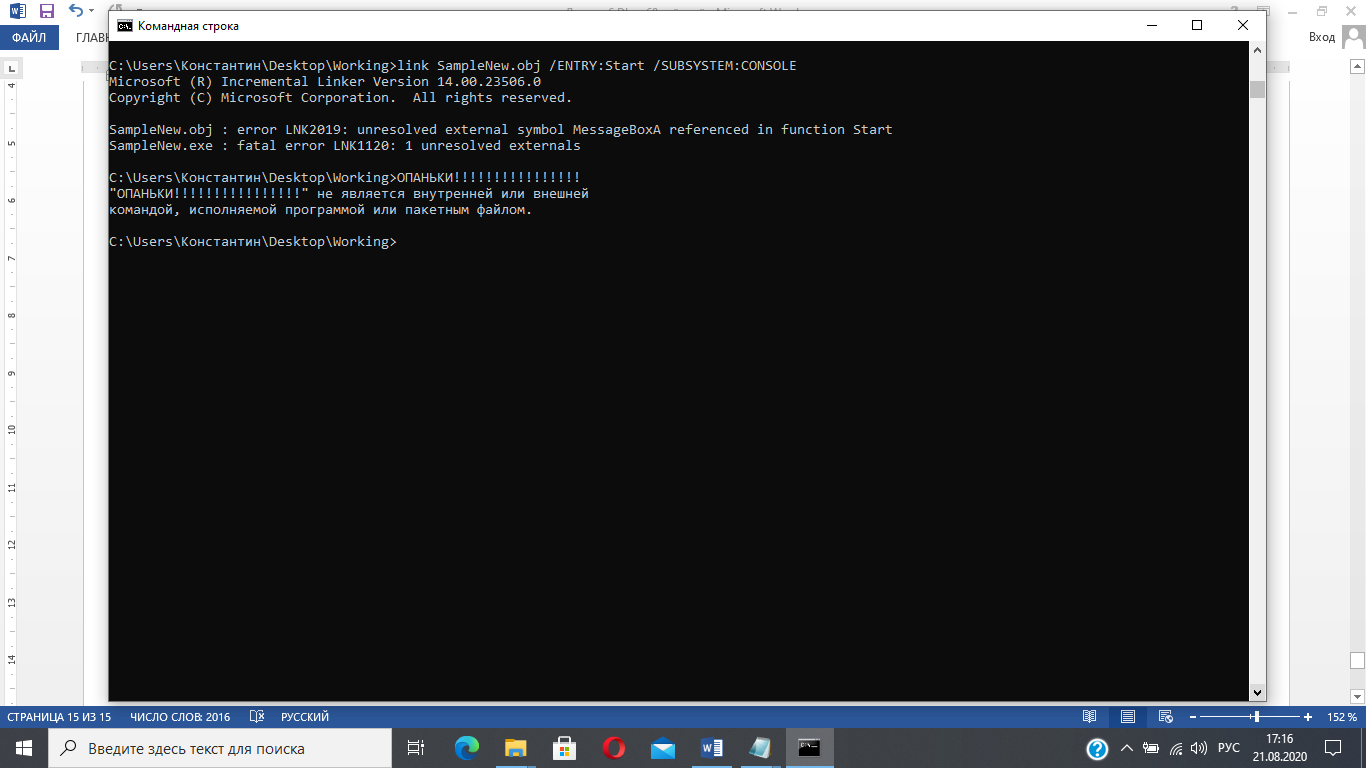


**Теперича опять этот LINK пишет Fatal error: a subsystem can’t be inferred and must be defined!!!!**

**Опять смотрим на опции LINK и ищем, что это за Subsystem! Находим, что есть про SUBSYSTEM.**

**Находим много чего интересного:**

**/SUBSYSTEM:{BOOT\_APPLICATION или CONSOLE … или WINDOWS…}. Я специально перечислил только те слова-опции, которые более-менее нам известны – типа CONSOLE или WINDOWS. Правда, возникает вопрос, что выбрать из этих трёх опций. С чего начнём? C любимой CONSOLE, хотя никто из Вас до конца и не знает, что это такое. Пробуем.**

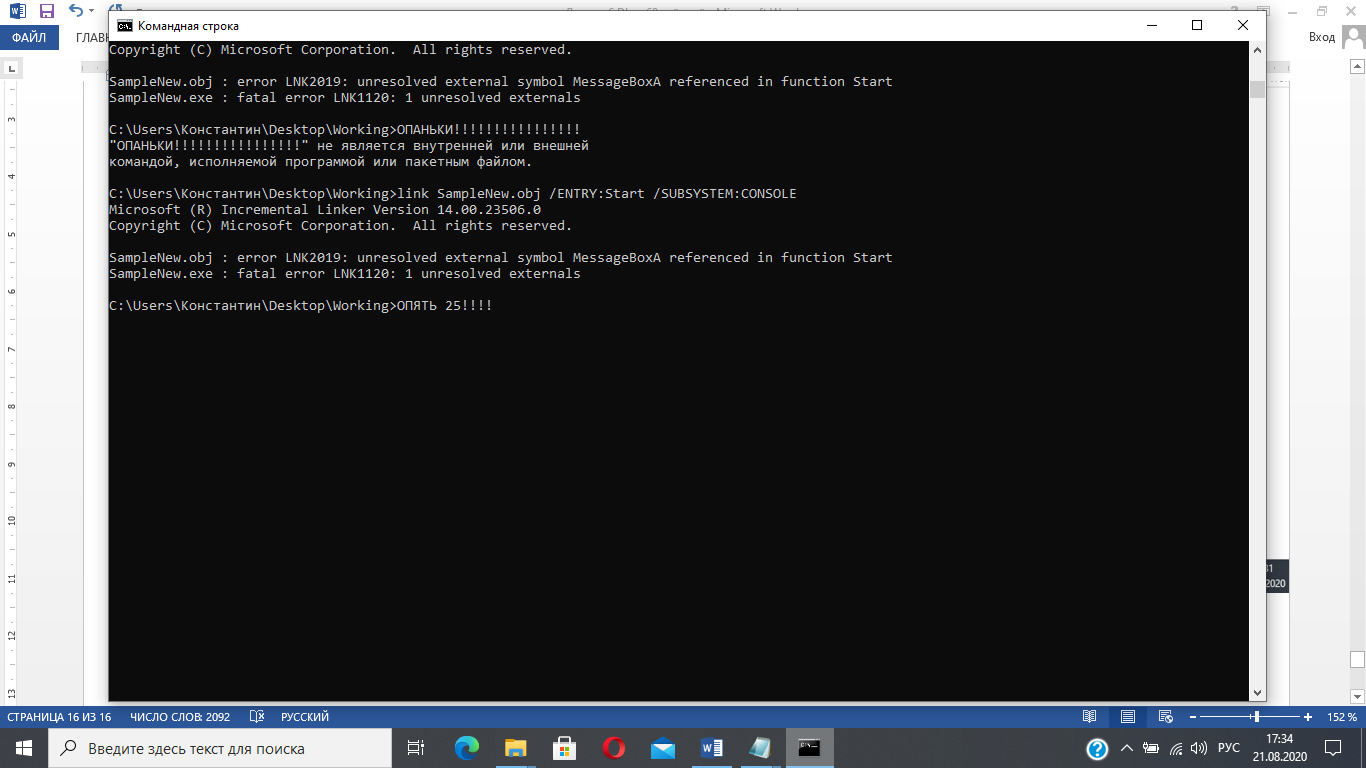


**Опять двадцать пять! Теперь этому LINKу не хватает MessageBoxA…. И снова, другого выхода-пути нет, как только пытаться копать головой в документации docs.microsoft.com. Очень похоже, что этот MessageBoxA где-то «сидит» себе тихонько в каком-нибудь LIBe. (Слово «библиотека» в контексте вычислительной машины произносить ЗАПРЕЩАЕТСЯ!!).**

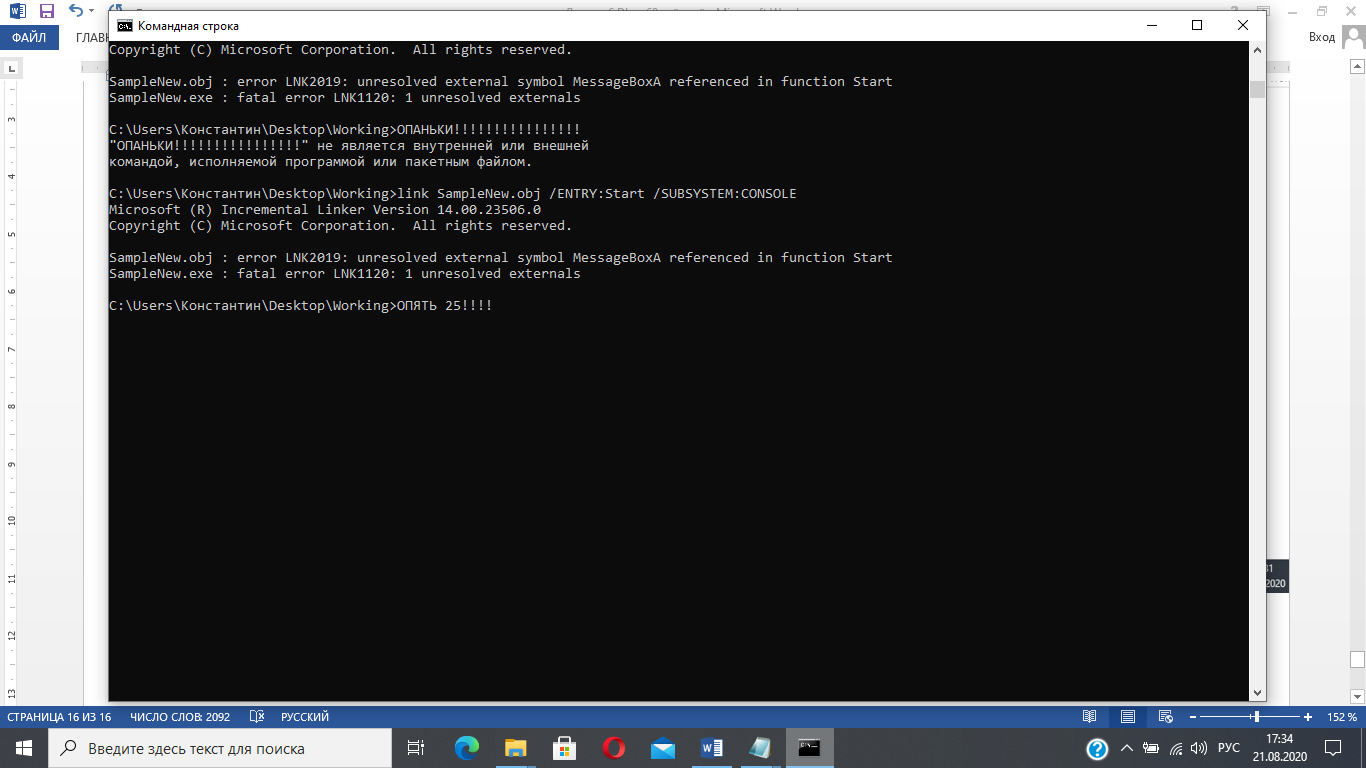
**Покопав отсюда и пока не найдёте, что MessageBoxA «сидит» в файле: User32.lib. Кольнуло: «Почему 32, а не 64?». Тем не менее – ищем User32.lib и «качаем» в нашу папку Working (Рис. 6.14.).**



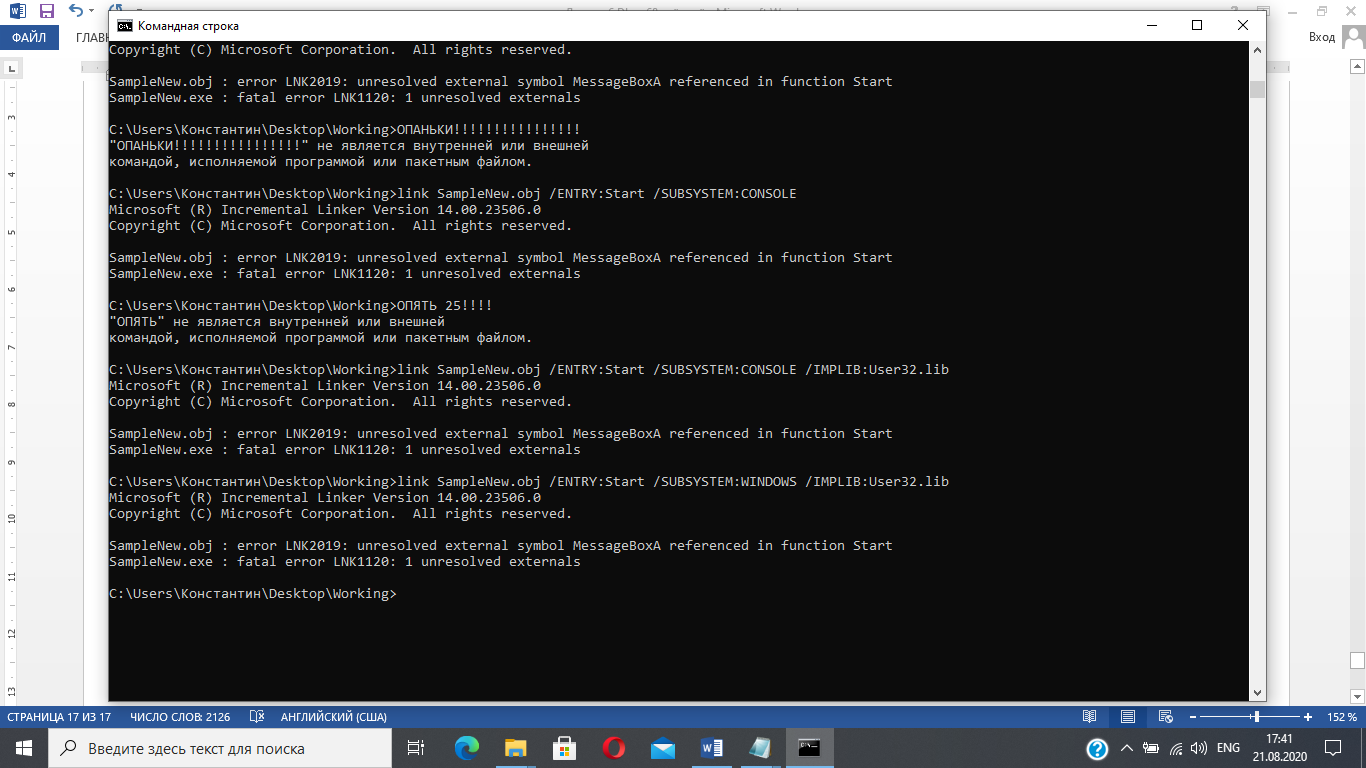
**И снова пробуем:**



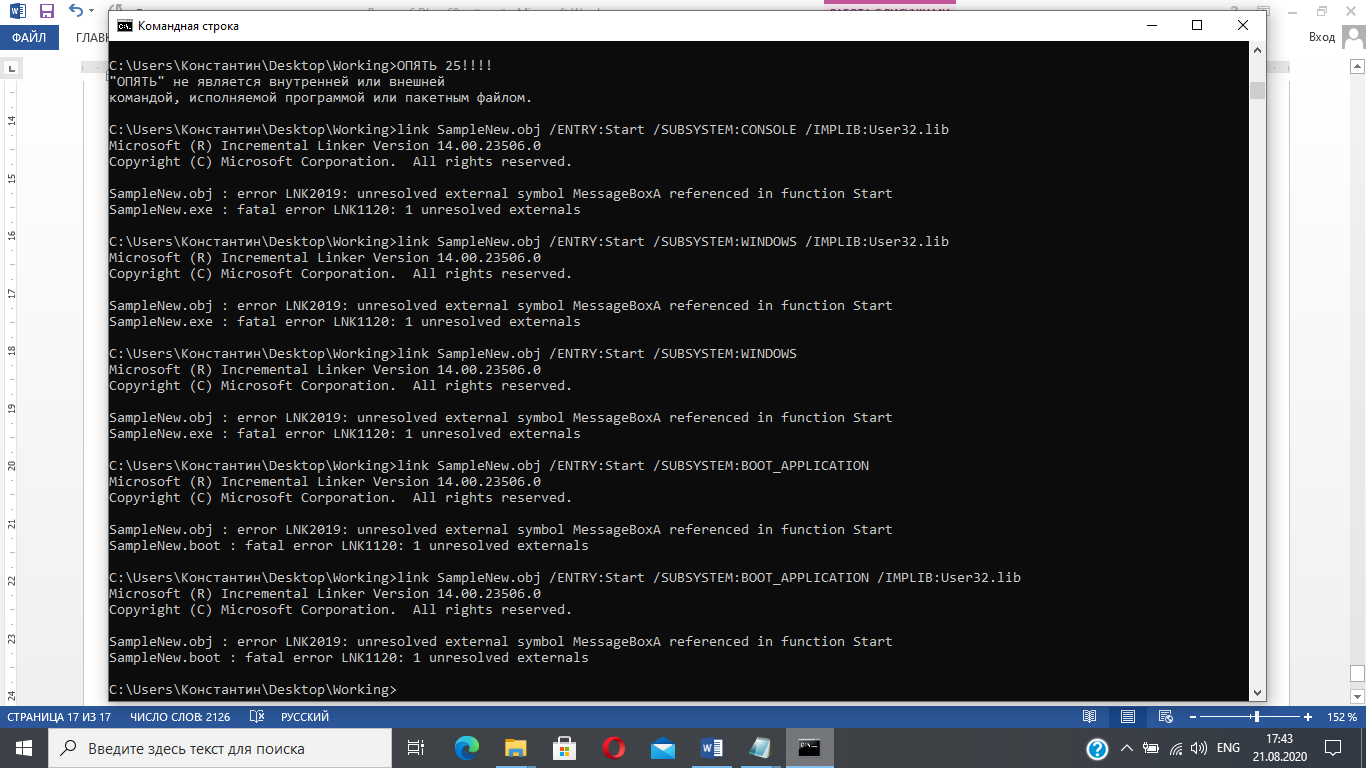
**LIBа недостаточно?! Нужно ещё указать, что за LIB нужон? Или CCONSOLE – не наша тема – система. Лезем снова в опции…. Там есть какая-то опция IMPLIB. Попробуем сначала это. Давайте….**



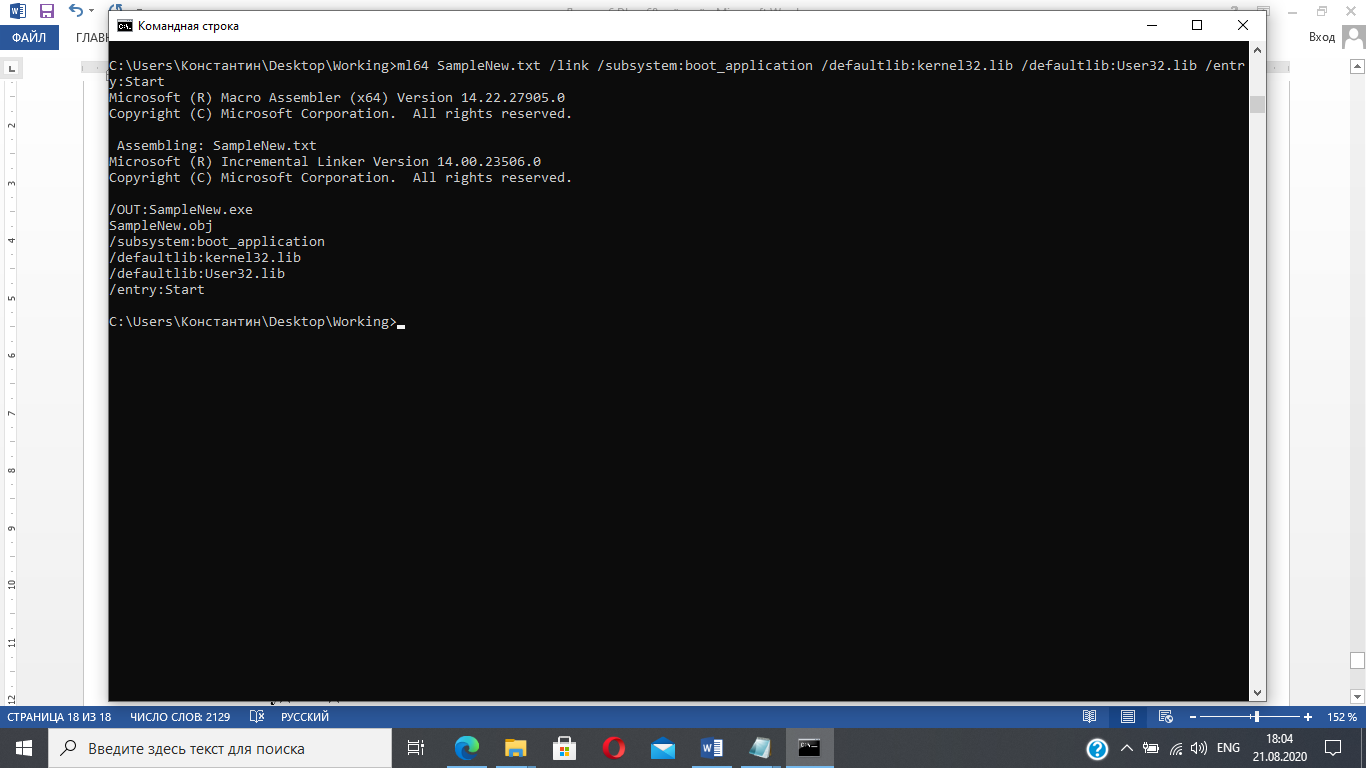
**Без изменений…. Тупик? Идём в Windows.**



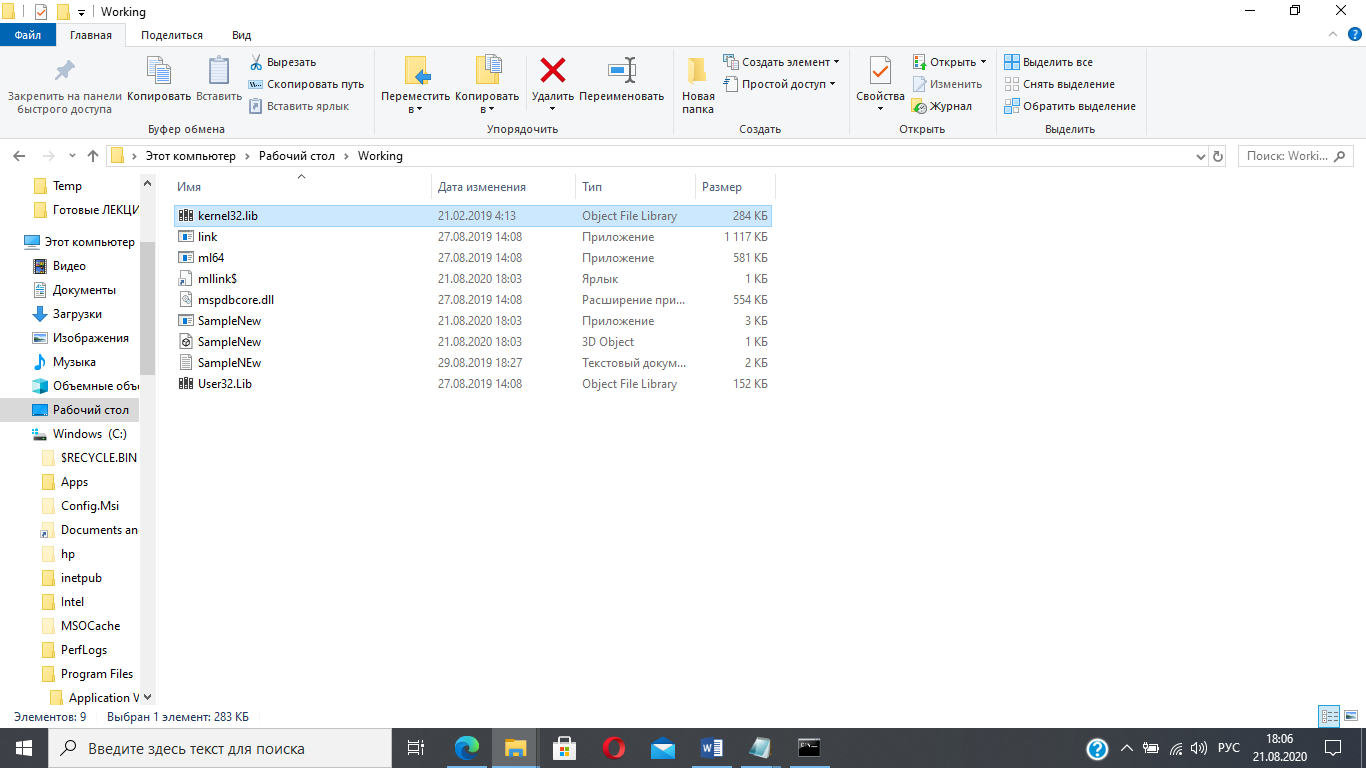
**ТУПИК!!!**



**Куда кидаться? «Покидавшись» часа два, получаем, что для получения исполнимого файла нужно набрать:**



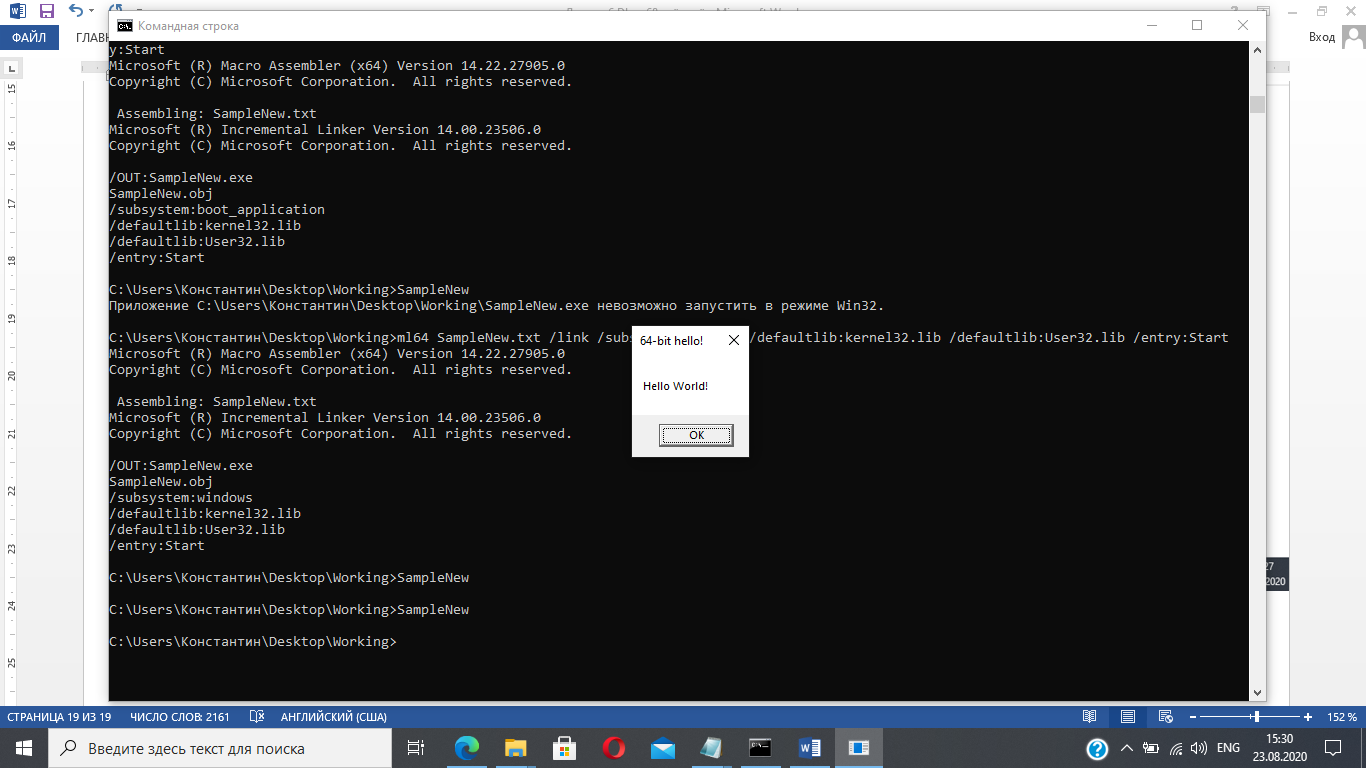
**Чтобы в папке Working получилась такая картинка:**



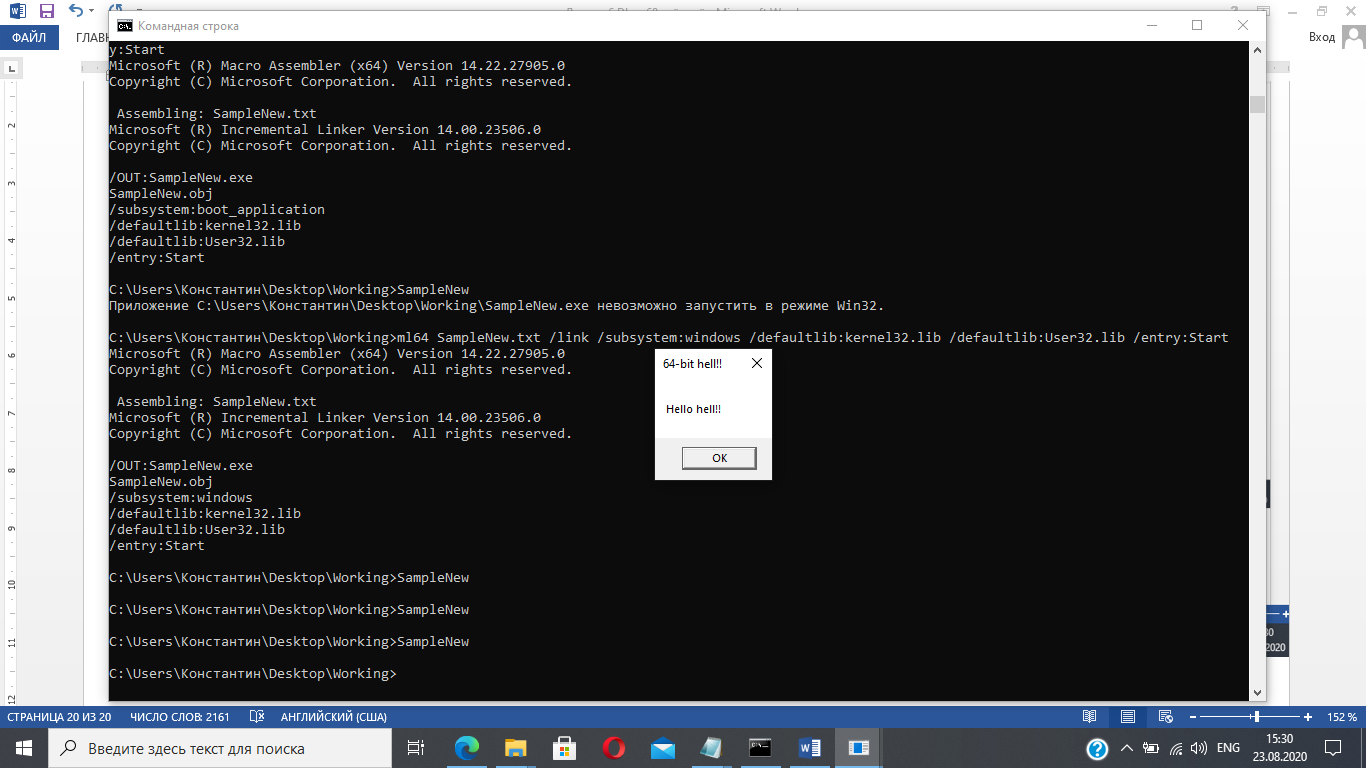
**Заметьте, что появился файл SampleNew!!! Представляющий собой исполнимый файл, который мы попробуем запустить:**



**Наконец-то?!!!!!!!!!!!!!!!!!!!!!!!!!!**



**ГЛУХО, как в танке. Пробуем ещё.**



**УРА! Я сделал это! Программу для 64 – разрядной ЭВМ с использованием ml64.exe для трансляции, LINK.exe для линкования и средств ОПЕРАЦИОННОЙ ОБОЛОЧКИ Windows”10 (средств «командной строки», эмулирующих реальную командную строку ОПЕРАЦИОННОЙ СИСТЕМЫ без названия)!!!!! А ВАМ СЛАБО?**