**ПЛАН ЗАНЯТИЯ**

**Дисциплина:** МДК.01.04 Системное программирование

**Преподаватель:** Галузин А.Б.

**Курс:** 4

**Группа:** П-40

**Специальность:** 09.02.07 Информационные системы и программирование

**Дата:** 26.09.25

**Время проведения:** 2 пара

**Вид занятия** практическое занятие

**Литература**

**Интернет-ресурсы:**

Assembler. Установка интерпретатора и запуск первой программы через DOSBox <https://habr.com/ru/articles/564898/>

Запуск DosBox <https://youtu.be/LByZ9bIQygA>

Урок по Assembler. #1 Настройка DosBox под TurboAssembler(tasm) <https://youtu.be/AWiIUuxur4U>

**ЗАДАНИЕ**: описать ход выполнения работы, включая текст программы, ответить на контрольные вопросы.

**Практическая работа №5**

**Тема**: **Создание и запуск программы на ассемблере**

**Цель**: Научиться создавать и запускать простые программы на ассемблере.

**Оборудование**: ПК. Программа Turbo Assembler 5.0.

1. **Инструктаж по ТБ при работе с ПК.**
2. **Методические рекомендации.**

**Процесс разработки программы**

Для 32-х разрядного процессора можно скачать пакет Tasm, который включает в том числе файлы tasm.exe, tlink.exe, td.exe. Для 64-х разрядного процессора следует установить виртуальную машину. В нашей работе рассмотрим пример создания и запуска программы на ассемблере для 32-х разрядного процессора.

На рис. 1 приведена общую схему процесса разработки программы на ассемблере. На схеме выделены четыре этапа этого процесса. На первом этапе, когда вводится код программы, можно использовать любой текстовый редактор. В Windows таким редактором может быть Блокнот (Notepad). При выборе редактора нужно учитывать, что он не должен вставлять «посторонних » символов (специальных символов форматирования). Созданный с помощью текстового редактора файл должен иметь расширение .asm.

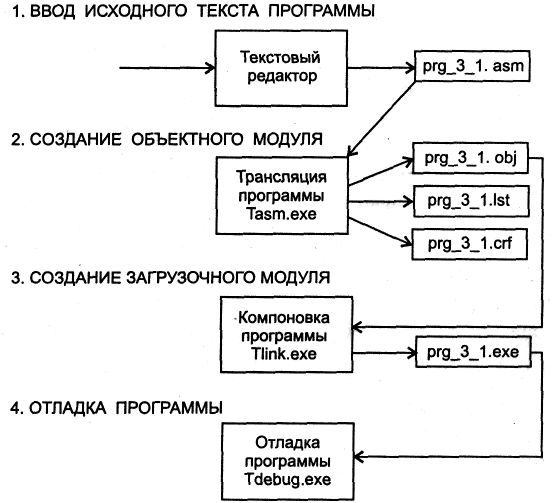


Рис. 1. Схема процесса разработки программы на ассемблере

**Трансляция программы**

Следующий шаг на пути создания выполняемого модуля — трансляция программы. Для трансляции нужен подготовленный и записанный на диск начальный текст программы. На этапе трансляции решается несколько задач:

* перевод команд ассемблера в соответствующие машинные команды;
* построение таблицы символов;
* расширение макросов;
* формирование файла листинга и объектного модуля.

Программа, которая реализует эти задачи, называется ***ассемблером***. Итог работы ассемблера — два файла: файл объектного модуля и файл листинга.

***Объектный модуль*** включает представление начальной программы в машинных кодах и некоторую другую информацию, необходимую для отладки и компоновки его с другими модулями. При использовании пакета TASM получения объектного модуля начального файла проводится программой (ассемблером) tasm.exe. Формат командной строки для запуска tasm.exe следующий:

TASM [ключи] имя\_исходного\_файла [,имя\_объектного\_файла]

[,имя\_файла\_листинга] [,имя\_файла\_перекрестных\_ссылок]

Обязательным аргументом командной строки является имя начального файла. Этот файл должен находиться на диске и обязательно иметь расширение .asm. За именем начального файла через запятую могут следовать необязательные параметры, которые помечают имена объектного файла, файла листинга и файла перекрестных ссылок. Если не задать их, то соответствующие файлы просто не будут созданы.

Если же их нужно создать, то необходимо учитывать некоторые моменты:

* Если имена объектного файла, файла листинга и файла перекрестных ссылок должны совпадать с именем начального файла (самый типичный случай), то нужно просто поставить запятые вместо имен этих файлов:

tasm.exe prgl

В результате будут созданные файлы с одинаковыми именами и разными расширениями, как показано на рис.1 для второго этапа.

* Если имена объектного файла, файла листинга и/или файла перекрестных ссылок не должны совпадать с именем начального файла, то нужно в командной строке указать имена нужных файлов в соответствующем порядке, например:

tasm.exe prgl, prg,\_list,

В результате на диске будут созданные файлы prg1.obj, prg\_list.lst, prg1.crf.

* Если нужно выборочное создание файлов, то вместо ненужных файлов необходимо подставить параметр nul.

Например: tasm.exe prg1,, nul,

В результате на диске будут созданные файлы prgl.obj, prg1.crf.

Необязательный аргумент [ключи] позволяет задавать режим роботы транслятора TASM.

Для устранения ошибок нужно определить место их возникновения и проанализировать ситуацию. Местоположение ошибки легко определяется по значению в скобках в сообщении об ошибке. Это значение является номером ошибочной строки. Запомнив его, нужно перейти в файл с начальной программой и по номеру строки найти место ошибки. Для локализации ошибок более целесообразно использовать информацию из специального создаваемого транслятором файла *листинга. Этот* файл имеет расширение .lst, а его имя определяется в соответствии с рассмотренными выше соглашениями.

Файл *листинга* содержит код ассемблера начальной программы, а также расширенную информацию об этом коде. Для каждой команды ассемблера указываются ее машинный (объектный) код и смещение в кодовом сегменте. Кроме того, в конце *листинга* TASM формирует таблицы с информацией о метках и сегментах, используемых в программе. Если есть ошибки или сомнительные участки кода, то TASM включает в конец *листинга* сообщения о них. Если сравнить их с сообщениями, которые выводятся на экран, то видно, что они совпадают. Кроме того, что очень удобно, эти же сообщения включаются в текст *листинга* непосредственно после ошибочной строки.

***Строки в файле*** *листинга* ***имеют следующий формат:***

глубина\_влажения номер\_строки смещение машинный\_код исходный\_код

где:

* **глубина\_** **влажения** — уровень вложенности файлов или макрокоманд в файле.
* **номер\_строки** — номер строки в файле *листинга*. Номера строк *листинга* используются для локализации ошибок и формирования ***таблицы перекрестных ссылок****.*
* **Смещение *-*** смещение в байтах текущей команды относительно начала сегмента кода. Его называют также счетчиком адреса. Величину смещение вычисляет транслятор для адресации в сегменте кода.
* **машинный\_код** *—* машинное представление команды ассемблера, представленной далее в этой строке полем исходный\_код.
* ***начальный\_код*** *—* строка кода из начального файла*.*

О нормальном окончании процесса трансляции можно судить по отсутствию строк с сообщениями об ошибках и предупреждениях.

Изучая внимательно файл *листинга*, следует обратить внимание на то, что не все строки начальной программы имеют соответствующий машинный код. Начальный файл на ассемблере в общем случае может содержать конструкции следующих типов:

* ***команды ассемблера*** *—* конструкции, которым отвечают машинные команды;
* ***директивы ассемблера*** *—* конструкции, которые не генерируют машинных команд, а являются указаниями транслятору на выполнение некоторых действий или служат для задания режима его работы;
* ***макрокоманды*** *—* конструкции, которые, будучи представленные одной строкой в начальном файле программы, после обработки транслятором генерируют в объектном модуле последовательность команд, директив или макрокоманд ассемблера.

**Компоновка программы**

После устранения ошибок и получения объектного модуля можно приступать к следующему этапу — создание исполняемого (загрузочного) модуля, или, как еще называют этот процесс, к *компоновке программы.* Главная цель этого этапа — превратить код и данные в объектных файлах в их *перемещаемое исполняемое отображение.* Раздел процесса создания исполняемого модуля на два *этапа* — трансляция и компоновка сделаны преднамеренно, чтобы можно было объединять вместе модули, написанные на одном и том же или на разных языках. Формат объектного файла позволяет при определенных условиях объединить несколько начальных модулей, которые отдельно оттранслировали, в один модуль. При этом в функции компоновщика входит разрешение внешних ссылок (ссылок на внешние процедуры и переменные) в этих модулях. Результатом работы компоновщика является создание загрузочного файла с расширением .ехе. После этого операционная система может загрузить такой файл в память и выполнить его.

Полный формат командной строки для запуска компоновщика достаточно сложный, мы в основном будем использовать упрощенный формат:

TLINK [ключи] перечень\_объектных\_файлов [,имя\_загрузочного\_модуля] [,имя \_файлу\_карты] [,имя \_файла\_библиотеки] [,имя \_файла\_определения] [,имя \_файла\_ ресурсов]

Параметры командной строки для запуска компоновщика:

* **ключи** — необязательные параметры, которые управляют работой компоновщика, Каждому ключу должен предшествовать символ - (дефис) или / (слеш). При задании имен ключей имеет значение регистр символов.
* **перечень\_объектных\_файлов** — обязательный параметр, что содержит список компонуемых файлов с расширением .obj. Файлы должны быть разделены пропусками или знаком + (плюс), например:

tlink /v prog + mdf + fdr

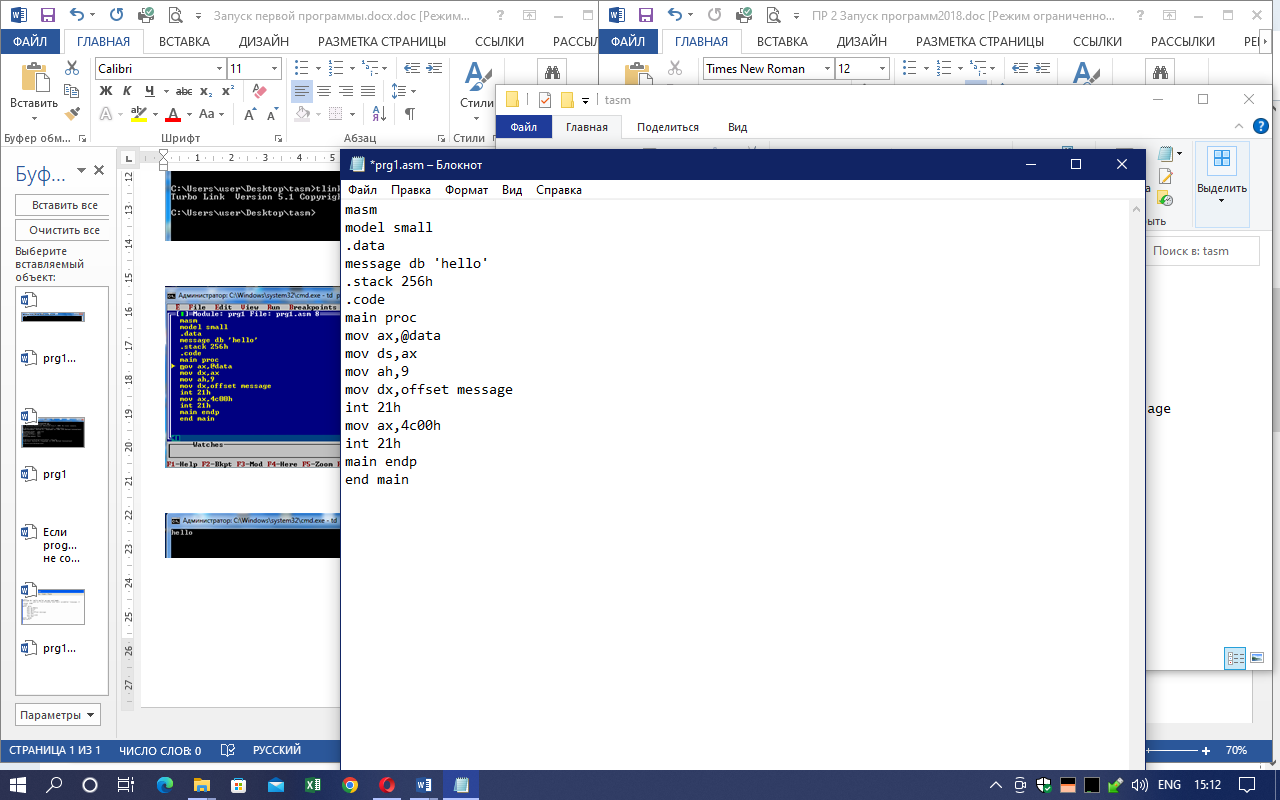
При необходимости имена файлов обеспечивают указанием пути к ним.

* **имя\_загрузочного\_модуля** — необязательный параметр, что помечает имя формируемого загрузочного модуля.
* **имя \_файлу\_карты** — необязательный параметр, наличие которого обязывает компоновщика создать специальный файл с картой загрузки. В ней перечисляются имена, адреса загрузки и размеры всех сегментов, которые входят в программу.
* **имя \_файла\_библиотеки** — необязательный параметр, который является путем к файлу библиотеки (.lib). Этот файл создается и обслуживается специальной утилитой tlib.exe пакета TASM. Утилита позволяет объединить часто используемые подпрограммы в виде объектных модулей в один файл
* **имя \_файла\_определения** — необязательный параметр, который является путем к|до| файлу определений (.def). Этот файл используется при компоновке Windows-дополнений|.
* **имя \_файла\_ ресурсов** — необязательный параметр, который является путем к файлу с ресурсами Windows-дополнения (.res). Этот файл используется при компоновке Windows-дополнений .

**Ход работы**

Перед работой с программой tasm.exe желательно провести некоторые подготовительные операции. После установки пакета TASM в каталоге \TASM\BIN, где находится файл tasm.exe, присутствующее большое количество файлов. Можно запустить программу tasm.exe прямо отсюда, но тогда созданы ею файлы объектного кода, *листинга* и перекрестных ссылок тоже окажутся в этом каталоге. Во избежание подобной ситуации, рекомендуется выполнить описанную дальше процедуру.

1. Создайте в каталоге \TASM вложенные каталоги . .\WORK и . .\PROGRAM. Каталог  
   ..\PROGRAM будет использоваться для хранения отлаженных кодов программ и их исполняемых модулей (файлов с расширением .ехе). Каталог ..\WORK станет рабочим — в нем будут находиться необходимые для получения исполняемого модуля файлы из пакета транслятора TASM и файл начального модуля, с которым в данный момент ведется работа. После устранения ошибок в начальном модуле его вместе с исполняемым модулем можно будет переписать в каталог . .\PROGRAM, а из каталога. .\WORK удалить все ненужные файлы, чтобы подготовить его для работы со следующим начальным модулем на ассемблере. Таким образом, в каталоге ..\WORK всегда будет находиться рабочая версия программы, а в каталоге ..\PROGRAM — отлаженная версия.
2. Поместите в каталог. .\WORK файлы tasm.exe, tlink.exe и rtm.exe. Если в дальнейшем в каталоге . .\WORK не окажется каких-то рабочих файлов вашей программы, программы tasm.exe и tlink.exe выдадут соответствующее сообщение.
3. Создайте в текстовом редакторе файл prg1.asm



1. Поместите в каталог. .\WORK файл prg1.asm.

5. После всех этих действий можно начинать работу.

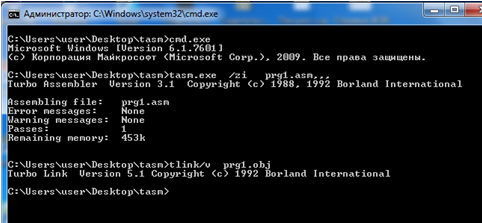
Для этого следует перейти в каталог . .\WORK и инициировать трансляцию программы prg1.asm командной строкой вида

tasm.exe /zi prg1.asm,,,

6. Если prog.asm не содержит ошибок, то в результате создается файл prg1.OBJ, который нужно скомпоновать при помощи компоновщика tlink.exe:

tlink /v prg1. obj

Если компоновка прошла успешно, на экране высветится:



Запустить полученный в ходе компоновки файл prg1.exe можно командой:

tdprg1.exe

В результате ее выполнения получим:



**Контрольные вопросы.**

1. Какие этапы проходит разработка программы на ассемблере?
2. Что происходит на этапе трансляции?
3. Что собой представляет объектный модуль?
4. Какая информация находится в файле *листинга*?
5. Какой формат имеют строки в файле *листинга*?
6. Какие функции выполняет компоновщик программ?