**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:** 24.09.24

**Время проведения:** 11.50-13.25, 3 пара

**Тема:** Процесс разработки программ на ассемблере.

**Цель занятия:**

**дидактическая:** изучить процесс разработки программ на ассемблере

**развивающая**: развивать абстрактное мышление, логику

**Вид занятия** лекция

**Литература**

Юров В.И. Assembler. Учебник для вузов. - 2-ое изд. – СПб.: Питер, 2003, стр. 128.

**Интернет-ресурсы:**

https://studopedia.su/6\_40462\_etapi-razrabotki-programmi-na-yazike-assemblera.html

**ЗАДАНИЕ**: законспектировать лекцию с учетом контрольных вопросов.

**КОНСПЕКТ ЛЕКЦИИ**

План

1. Жизненный цикл программы.
2. Процесс разработки программы.

**1. Жизненный цикл программы**

Процесс разработки программы на ассемблере, включая постановку задачи, получение первых результатов и дальнейшее сопровождение программы, мало чем отличается от традиционного подхода с использованием любого из существующих языков программирования. После адаптации основных положений этого процесса к особенностям ассемблера получится вполне стандартный перечень работ.

1. Постановка и формулировка задачи:

a. изучение предметной области и сбор материала в проблемно-ориентированном контексте;

b. определение назначения программы, выработка требований к ней и представление требований, если возможно, в формализованном виде;

c. формулирование требований к представлению исходных данных и выходных результатов;

d. определение структур входных и выходных данных;

f. формирование ограничений и допущений на исходные и выходные данные.

2. Этап проектирования:

a. формирование «ассемблерной» модели задачи; выбор метода реализации задачи; разработка алгоритма реализации задачи;

b. разработка структуры программы в соответствии с выбранной моделью памяти.

3. Этап кодирования:

a. уточнение структуры входных и выходных данных и определение ассемблерного формата их представления;

b. программирование задачи;

c. комментирование текста и составление предварительного описания программы.

4. Этап отладки и тестирования:

a. составление тестов для проверки работоспособности программы;

b. обнаружение, локализация и устранение в программе ошибок, выявленных в тестах;

c. корректировка кода программы и ее описания.

5. Этап эксплуатации и сопровождения:

a. настройка программы на конкретные условия использования;

b. обучение пользователей работе с программой;

c. организация сбора сведений о сбоях в работе программы, ошибках в выходных данных, пожеланиях по улучшению интерфейса и удобства работы с программой;

d. модификация программы с целью устранения выявленных ошибок и, при необходимости, изменения ее функциональных возможностей.

Порядок и объем работ в приведенном перечне укладываются в понятие жизненного цикла программы на ассемблере. На практике к порядку применения и полноте выполнения перечисленных этапов нужно подходить разумно и творчески. Многое определяется особенностями конкретной задачи, ее назначением, объемами кода и обрабатываемых данных, другими характеристиками исходной задачи.

Некоторые из этих этапов могут либо выполняться одновременно с другими, либо вовсе отсутствовать.

Главная цель формирования подобного списка работ — в том, чтобы изначально упорядочить процесс создания нового программного продукта с сохранением концептуальной целостности постановки задачи и исключением анархии в процессе разработки.

**2. Процесс разработки программы**

Общая схема процесса разработки программы на ассемблере включает 4 этапа (рис. 1):

* Ввод исходного текста программы.
* Создание объектного модуля.
* Создание загрузочного модуля.
* Отладка программы..

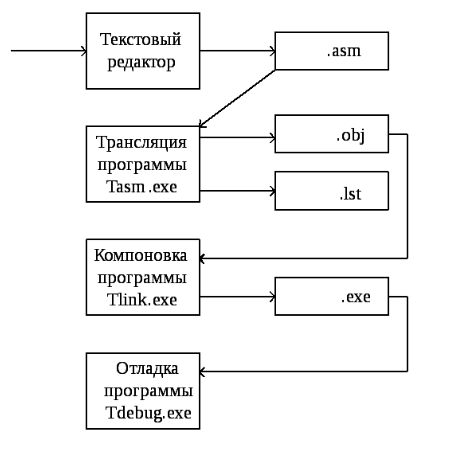


Рис.1

**Ввод исходного текста программы**

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). С этой точки зрения Microsoft Word в качестве основного редактора ассемблерных программ не годится. Созданный с помощью текстового редактора файл должен иметь расширение .asm.

Для выполнения остальных этапов разработки требуется специализированные программные средства из пакета МASM или TASM.

Рассмотрим процесс разработки программ с помощью пакета TASM. В принципе все пакеты ассемблера выполняют практически одну работу, но по-разному, например, маскируют ее с помощью интегрированной среды или объединяют некоторые этапы разработки.

**Трансляция программы**

Следующий шаг на пути создания исполняемого модуля - *трансляция* программы. На этапе трансляции осуществляется перевод команд ассемблера в соответствующие машинные команды. В результате трансляции решается несколько задач:

* перевод команд ассемблера в соответствующие машинные команды;
* построение таблицы символов (структуры данных в компиляторе, где хранятся все идентификаторы, используемые в программе, вместе с их типом, областью действия и ячейками памяти);
* расширение макросов (макрос — это функция, вызываемая компилятором во время компиляции программы. Её цель — изменение имеющегося кода или генерация нового кода);
* формируются файлы объектного модуля и файл листинга.

Программа, которая реализует эти задачи, называется ассемблером. Итог работы ассемблера - два файла: файл объектного модуля и файл листинга. Объектный модуль включает в себя представление исходной программы в машинных кодах и некоторую другую информацию, необходимую для отладки и компоновки его с другими модулями.

При использовании пакета TASM получение объектного модуля исходного файла производится 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]

tasm.exe prg\_6\_l , , ,

В результате будут созданы файлы с одинаковыми именами и разными расширениями (рис. 1).

tasm.exe prg\_6\_l , ,prg\_list,

В результате будут созданы файлы prg\_6\_1.obj, prg\_list.lst, prg\_6\_1.crf

tasm.exe prg\_6\_l , , nul ,

В результате будут созданы файлы prg\_6\_1.obj, prg\_6\_1.crf

Необязательный аргумент [ключи]позволяет задавать режим работы транслятора TASM

Ключ /z разрешает вывод на экран строк исходного текста программы, в которых ассемблер обнаружил ошибки.

Ключ /zi управляет включением в объектный файл номеров строк исходной программы и другой информации, не требуемой при выполнении программы, но используемой отладчиком.

**Компоновка программы**

После устранения ошибок и получения объектного модуля можно приступать к следующему этапу - созданию исполняемого (загрузочного) модуля, или *компоновке* программы. Главная цель этого этапа - преобразовать код и данные в объектных файлах в их *перемещаемое выполняемое отображени*е. Разделение процесса создания исполняемого модуля на два этапа, трансляцию и компоновку, позволяет объединять модули, написанные на одном и том же или на разных языках. Результатом работы компоновщика является создание загрузочного файла с расширением .exe.

Полный формат командной строки для запуска компоновщика:

TLINK [ключи] список объектных файлов [,имя загрузочного модуля] [,имя файла карты] [,имя файла библиотеки] [,имя файла определений] [,имя ресурсного файла]

После этого операционная система может загрузить такой файл и выполнить его.

Устранение синтаксических ошибок еще не гарантирует того, что программа будет хотя бы запускаться, не говоря уже о правильности работы. Поэтому обязательным этапом процесса разработки является отладка.

**Отладка программы**

На этапе отладки в соответствии с алгоритмом проверяется правильность функционирования как отдельных фрагментов кода, так и программы в целом.

Специфика программ на ассемблере состоит в том, что они интенсивно работают с аппаратными ресурсами компьютера. Это обстоятельство заставляет программиста постоянно отслеживать содержимое определенных регистров и областей памяти. Естественно, что человеку трудно следить за этой информацией с большой степенью детализации. Поэтому для локализации логических ошибок в программах используют специальный тип программного обеспечения - программные отладчики.

Отладчики бывают двух типов:

* интегрированные — отладчик реализован в виде интегрированной среды типа среды для языков Turbo Pascal, Quick С и т.д.;
* автономные — отладчик представляет собой отдельную программу.

Из-за того, что ассемблер не имеет своей интегрированной среды, для отладки написанных на нем программ используют автономные отладчики. К настоящему времени разработано большое количество таких отладчиков. В общем случае с помощью автономного отладчика можно исследовать работу любой программы, для которой создан исполняемый модуль, независимо от того, на каком языке был написан его исходный текст.

Запуск отладчика удобнее производить из командной строки с указанием исполняемого модуля отлаживаемой программы:

td имя\_исполняемого\_модуля

**Контрольные вопросы**

1. Опишите подробно схему процесса разработки программы на ассемблере.
2. Как ввести исходный текст программы?
3. Что такое трансляция и какие задачи решаются на этапе трансляции?
4. Запишите полный формат командной строки для запуска компоновщика.
5. Как запустить отладчик программы?