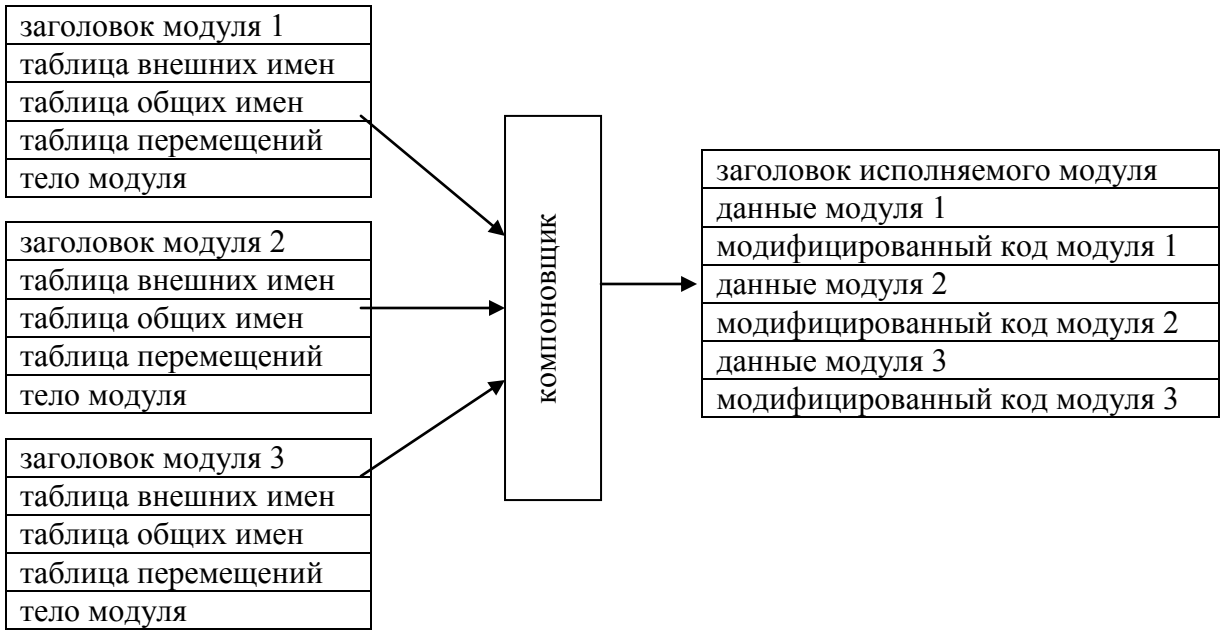
18. Логика работы программ-компоновщиков

Цель компоновщика – объединение отдельных объектных модулей в единый исполняемый модуль. На вход компоновщика поступает поток объектных модулей, адреса в каждом из которых настроены на нулевое значение. Компоновщик должен выполнить следующие действия:

1. собрать вместе данные и команды из каждого отдельного модуля с настройкой операндов команд на единый нулевой начальный адрес
2. обработать перекрестные связи между модулями и заменить нулевые адреса в операндах, где использовались внешние имена
3. создать выходной модуль в исполняемом формате

Тем самым, компоновщик от набора отдельных адресных пространств модулей должен перейти к единому адресному пространству, в котором должны располагаться данные и команды всех модулей.

Будем считать, что все входные модули имеют одинаковую структуру и включают в себя заголовок, таблицу внешних имен, таблицу общих имен, таблицу перемещений и тело с данными и командами.



Входные объектные модули обрабатываются последовательно, по мере поступления на вход компоновщика, поэтому в процессе их обработки может возникнуть ситуация, похожая на ситуацию ссылки вперед при

ассемблировании. В отличие от ассемблирования, эта ситуация носит межмодульный характер, т.е. ссылки вперед могут возникнуть между модулями. Более подробно эту ситуацию можно объяснить следующим образом. Для реализации второго действия в приведенном выше списке компоновщик должен сопоставить каждому внешнему имени адрес в общем адресном пространстве создаваемого исполняемого модуля. Необходимую для этого информацию компоновщик должен извлечь из локальных ТОИ модулей, и при этом вполне возможна ситуация, когда в ТВИ текущего модуля встречается имя, которое в ТОИ предыдущих модулей еще не встречалось и поэтому считается неопределенным (адрес ему не назначен).

Для выхода из этой ситуации можно применить подход, аналогичный используемому при ассемблировании – реализовать компоновщик по более простой двухпроходной схеме или по более сложной однопроходной. Рассмотрим двухпроходную схему.

Основная цель первого прохода – построение глобальной ТОИ на основе локальных ТОИ входных модулей. Для перехода от локальных адресных пространств модулей к глобальному компоновщик использует внутреннюю переменную – адрес размещения модуля (АРМ). Эта переменная определяет размещение текущего модуля в общем адресном пространстве. Начальное значение АРМ равно нулю, и после обработки очередного модуля значение АРМ увеличивается на байтовый размер этого модуля. Размер модуля компоновщик извлекает из заголовка модуля. Текущее значение АРМ используется для корректировки адресов при переходе от локального адресного пространства к глобальному.

Более подробно алгоритм первого прохода можно представить следующим образом.

1. загрузить первый входной модуль и скопировать его данные и команды в промежуточный файл без каких-либо изменений
2. занести ТОИ модуля в глобальную ТОИ без каких-либо изменений
3. занести ТВИ модуля в глобальную ТВИ

4. увеличить значение АРМ на байтовый размер модуля, что тем самым определит адрес размещения в общем пространстве второго модуля
5. загрузить второй входной модуль и скопировать его данные в промежуточный файл
6. модифицировать команды второго модуля следующим образом:
    a. извлечь из таблицы перемещений адрес операнда и увеличить его на значение переменной АРМ (тем самым определяется адрес операнда в общем адресном пространстве)
    b. обратиться по вычисленному адресу и изменить находящийся там адрес за счет увеличения его на значение переменной АРМ
7. добавить ТОИ модуля в глобальную ТОИ с выполнением следующих действий:
    a. проверить каждое имя на уникальность с формированием при необходимости сообщения об ошибке
    b. увеличить связанный с именем адрес на значение переменной АРМ
8. добавить ТВИ модуля в глобальную ТВИ с увеличением адреса внешнего имени на значение переменной АРМ
9. увеличить значение АРМ на байтовый размер модуля и перейти к обработке следующего модуля, повторяя все действия с шага 5

В итоге, после первого прохода будет построена глобальная таблица общих имен, содержащая уникальные имена и назначенные им адреса в общем адресном пространстве. Кроме того, буде построена вспомогательная общая таблица внешних имен, в которой адреса подлежащих замене операндов будут привязаны к общему адресному пространству.

На втором проходе выполняется окончательная настройка операндов команд, а именно – замена нулевых адресов в тех операндах, где были использованы внешние имена. Для этого используются построенные на первом проходе глобальные таблицы общих и внешних имен. Более

конкретно, каждая запись в глобальной ТВИ просматривается и обрабатывается следующим образом:

- извлекается внешнее имя, которое отыскивается в глобальной ТОИ с формированием соответствующего результата (возможна ошибка типа “Неопределенное внешнее имя”)
- из глобальной ТОИ извлекается адрес, связанный с найденным именем
- из записи извлекается адрес изменяемого операнда (первое поле) и по этому адресу заносится найденный на предыдущем шаге адрес

Однопроходная схема реализуется аналогично ассемблирующей программе за счет усложнения структуры глобальной таблицы общих имен. При построении этой таблицы должны запоминаться адреса тех операндов, где используются неопределенные пока внешние имена. Для этого при обработке текущего модуля просматриваются обе его локальные таблицы – как обычно ТОИ и дополнительно ТВИ и все встречающиеся в них имена заносятся в глобальную ТОИ. При этом все неопределенные имена собираются во вспомогательные списки с запоминанием адресов операндов из ТВИ. Как только неопределенное имя распознается в ТОИ какого-либо модуля, вспомогательные списки просматриваются и операнды получают свое окончательное значение.

За счет небольшого изменения алгоритма компоновщик может выполнить реструктуризацию исполняемого модуля, собрав в его начале все данные, а уж потом – все команды.