Генерация кода

После того как синтаксис программы проанализирован последним шагом процесса компиляции является генерация объектного кода. Задача генератора кода – построение эквивалентной машинной программе по программе на входном языке. Обычно в качестве входного для генератора кода служит некоторое промежуточное представление программы. Но код может генерироваться и при обходе дерева разбора, построенного анализатором на предыдущих стадиях.

Генерация кода состоит из ряда специфических относительно независимых подзадач:

* Распределение памяти
* Выбор команд
* Генерация объектного или загрузочного модуля

Схема генератора кода зависит от формы промежуточного представления. Генерация кода из дерева грамматического разбора отличается от генерации кода из триад, а генерация кода из префиксной записи отличается от генерации кода из ориентированного графа.

Проверка синтаксической корректности и генерация кода требует знаний характеристики идентификаторов, имен и обозначений, которые используются в программе на входном языке. Такая информация занесена во все таблицы после анализа программы, и компилятор должен переходить к построению соответствующей программы в машинном коде. Фактически для получения машинного кода требуется 2 отдельных прохода:

1. Генерация промежуточного кода. Можно проектировать на различных уровнях. Иногда промежуточный код получают просто разбивая сложные структуры исходного языка на более удобные для обращения элементы, однако чаще в качестве промежуточного кода используют какой-либо обобщенный машинный код. Чаще всего это тетрады или триады.
2. Генерация машинного кода

Самым простым методом генерации кода является метод, который генерирует объектный код для каждого фрагмента программы как только распознан синтаксис этого фрагмента. Для генерации такого фрагмента необходим набор подпрограмм, соотвествующих каждому правилу и каждой альтернативе в правилах грамматики. В результате процесса грамматического разбора будет распознан текст программы, соответствующий некоторому правилу грамматики, вызывается подпрограмма соответствующая этому правилу. Большинство коммерческих компиляторов формируют объектную программу в виде объектного модуля, т.е. как последовательность записей, которые содержат информацию на машинном языке и возможности перемещения и связывания кода. Для формализованного анализа необходимо простроить формализованную модель ресурсов целевого компьютера. В состав которых входят:

* Модель информационных или запоминающих ресурсов
* Модель операционных ресурсов, эквивалентных операций ЯП в виде таблицы команд, которая включает коды операции и базовых модификаций адресов в качестве ключей и соответствующие машинные коды как характеристики поиска.

Общая схема распределения памяти может выглядеть следующим образом: каждая процедура, которая компилируется, имеет область данных в которую входят все внутренние переменные, фактические параметры и т.д. Элемент таблицы для каждой области данных содержит поле с ключом определяющий элемент как статический или динамический. Требуется, чтобы для таблицы символов можно было узнать размер области памяти нужной для каждой переменной.

Языка с нестрогими условиями декларации данных не вся информация известна к окончанию просмотра всей исходной программы. Поэтому для присваивания адресов после обычного анализа необходим второй просмотр. Во время первого основного просмотра компилятор генерирует машинные команды непосредственно из входного языка. Ссылки на переменные в машинных командах определяются указателями на соотвествующие им элементы таблицы символов во время второго просмотра переменным присваиваются адреса, которые запоминаются в элементах таблицы символов. В языках, которые требуют описание переменных до их использования распределение памяти выполняется семантическими программами, предназначенными для обработки декларации данных. В этом случае возможен однопросмотровый компилятор. Процесс генерации кода отдельных команд опирается на каноническое множество адресов и модификации команд. Во множество адресов включается как минимум:

1. Прямая адресация глобальных и динамических данных
2. Относительная адресация аргументов и локальных данных процедур и функций в стеке
3. Индексированная адресация глобальных и локальных данных

На основе допустимых операций и форматов адресов данных строится таблица, в которой аргументами являются последовательность команд и формы адресации сопоставимые с выборками из тетрадной формы представления команд и характеризующийся простыми правилами формирования адресов в объектном коде.

Относительный адрес любой входной точки относительно начала сегмента является константой. И для определения такого имени для компоновщика достаточно рядом с его символическим образом определить сегмент и относительный адрес. Словарь внешних символов определяет и идентифицирует кодовый сегмент объектной программы, внешние ссылки на программные сегменты, и другие входные точки, которым обращаются в данном модуле, но которые не включены в него. Объединение сегментов данных в большие физические сегменты требует возможности коррекции относительного адреса в уже сформированной команде назначение относительного смещения начала логического сегмента. Генерируемый код зависит от того на каком компьютере будет выполняется компилируемая программа так как большинство компиляторов генерирует коды непосредственно на машинном языке.