Тема 8. Синтаксически управляемая трансляция инструкции case

Рассмотрим следующую инструкцию **switch**.

switch E
begin
    case V_1 : S_1
    case V_2 : S_2
    …
    case V_{n-1} : S_{n-1}
    default : S_n
end

При использовании схемы синтаксически управляемой трансляции удобно транслировать эту конструкцию в промежуточный код, представленный на рис. 1.

```
            Код вычисления E в t
            goto test
L1:         Код для S1
            goto next
L2:         Код для S2
            goto next
            …
Ln–1:Код для Sn–1
            goto next
Ln:         Код для Sn
            goto next
test:       if t = V1 goto L1
            if t = V2 goto L2
            …
            if t = Vn–1 goto Ln–1
            goto Ln
next:
```

Рис. 1

Все проверки оказываются в конце, так что генератор кода может распознать многопутевое разветвление программы и сгенерировать для него эффективный код, основанный на одной из описанных технологий. Если мы создадим более простой промежуточный код, показанный на рис. 2, компилятор должен выполнить обширный анализ для поиска наиболее эффективной реализации. Заметим, что размещение кода ветвления в начале неудобно, поскольку тогда компилятор не может строить код для каждого S_i в момент его появления.

```
        Код вычисления E в t
        if t ≠ V1 goto L1
        Код для S1
        goto next
L1:     if t ≠ V2 goto L2
        Код для S2
        goto next
L2:
        …
Ln–2:if t ≠ Vn–1 goto Ln–1
        Код для Sn–1
        goto next
Ln–1:Код для Sn
next:
```

Рис. 2

Для трансляции (рис. 1) ключевого слова **switch** создаем две новые метки *test* и *next* и новую переменную *t*. Затем, по мере разбора выражения *E*, генерируется код для вычисления *E* в *t*. После обработки *E* генерируем переход **goto** *test*.

Затем, когда появляется ключевое слово **case**, создаем новую метку L_i. В очередь, используемую исключительно для хранения информации о значениях **case**, мы помещаем метку и значение V_i константы при **case**.

Обрабатываем каждую инструкцию **case** V_i: S_i путем генерации новой метки L_i, за которой следует код для S_i с последующим переходом **goto** *next*. После того как появится завершающее конструкцию ключевое слово **end**, мы готовы сгенерировать код ветвления. Считывая пары указатель-значение из очереди, мы можем генерировать последовательность трехадресных инструкций вида

case V_1 L_1
case V_2 L_2
…
case V_{n-1} L_{n-1}
case t L_n
Label *next*

где t – имя, хранящее значение селектора E, а L_n – метка инструкции по умолчанию. Трехадресная инструкция **case** V_i L_i является синонимом **if** $t = V_i$ **goto** L_i (рис. 17), однако использование **case** облегчает генератору целевого кода задачу нахождения потенциальных кандидатов для специальной обработки. На стадии генерации кода последовательность инструкций **case** может быть транслирована в n-путевое ветвление наиболее эффективного вида, в зависимости от количества значений и их размещения в небольшом диапазоне.