

Тема 6. Генерация промежуточного кода

6.4. Трансляция управляющих операторов

СУО для трансляции в трехадресный код основных управляющих операторов представлено в табл. 13. Для полноты и наглядности установки начальных значений наследуемых атрибутов в СУО добавлены и другие продукции.

Таблица 13

СУО для трансляции управляющих операторов

Продукция	Семантические правила
$P \rightarrow S$	$S.next := NewLabel()$ $P.code := S.code \parallel Label(S.next)$
$S \rightarrow id := E$	см. табл. 9
$S \rightarrow id := B$	см. табл. 12
$S \rightarrow \text{if } B \text{ then } S_1$	$B.true := NewLabel(); B.false := S.next$ $S_1.next := S.next$ $S.code := B.code \parallel Label(B.true) \parallel S_1.code$
$S \rightarrow \text{if } B \text{ then } S_1 \text{ else } S_2$	$B.true := NewLabel(); B.false := NewLabel()$ $S_1.next := S.next; S_2.next := S.next$ $S.code := B.code \parallel Label(B.true) \parallel S_1.code$ $\parallel Gen('goto' S.next) \parallel Label(B.false) \parallel S_2.code$
$S \rightarrow \text{while } B \text{ do } S_1$	$beg := NewLabel()$ $B.true := NewLabel(); B.false := S.next$ $S_1.next := beg$ $S.code := Label(beg) \parallel B.code$ $\parallel Label(B.true) \parallel S_1.code \parallel Gen('goto' beg)$
$S \rightarrow \text{repeat } S_1 \text{ until } B$	$B.true := S.next; B.false := NewLabel()$ $S.code := Label(B.false) \parallel S_1.code \parallel B.code$
$S \rightarrow S_1 ; S_2$	$S_1.next := NewLabel(); S_2.next := S.next$ $S.code := S_1.code \parallel Label(S_1.next) \parallel S_2.code$

Логическое выражение B имеет наследуемые атрибуты $B.true$ и $B.false$ (см. СУО в табл. 12). С этими атрибутами связаны метки, которым передается управление в случае истинности или ложности выражения B соответственно. Оператор S имеет наследуемый атрибут $S.next$, с которым связана метка, указывающая на команду, непосредственно следующую за кодом S .

Структура генерируемых этим СУО кодов показана на рис. 9. Для операторов **if-then** и **while** значения меток *B.false* и *S.next* совпадают, для оператора **repeat** совпадают значения меток *B.true* и *S.next*.

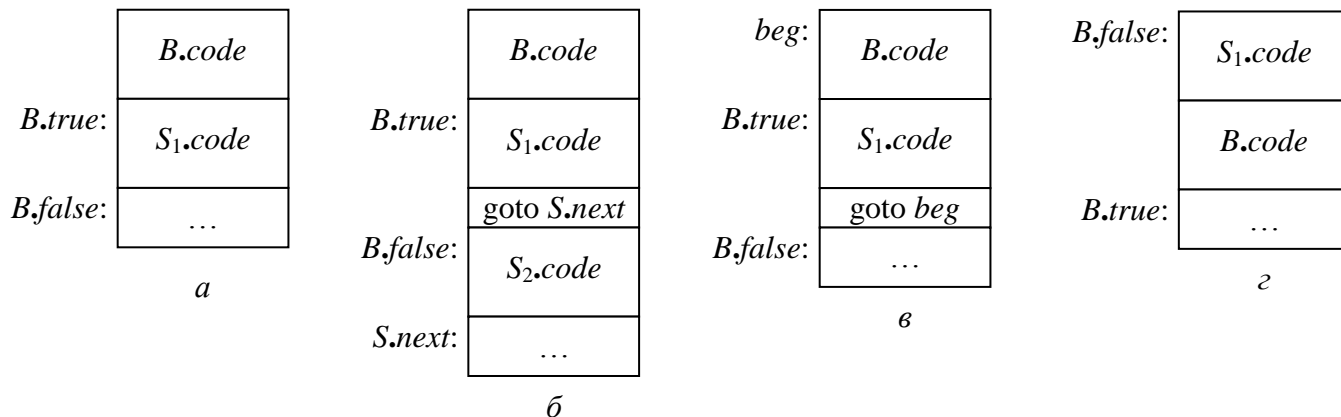


Рис. 9. Структура трехадресных кодов для управляющих операторов:
a – **if-then** (*B.false* = *S.next*); *б* – **if-then-else**; *в* – **while** (*B.false* = *S.next*); *г* – **repeat** (*B.true* = *S.next*)

В правилах для продукции $S \rightarrow \text{if } B \text{ then } S_1$ создается новая метка $B.true$, которая назначается первой трехадресной команде, генерируемой для оператора S_1 (рис. 9, а). Установкой атрибуту $B.false$ значения $S.next$ обеспечивается пропуск кода для оператора S_1 в случае ложности значения B . Очевидно, что непосредственно следующей за S команда с меткой $S.next$ является также непосредственно следующей за S_1 . Поэтому наследуемому атрибуту $S_1.next$ устанавливается значение (метка) $S.next$.

В правилах для оператора **if-then-else** (рис. 9, б) создаются новые метки $B.true$ и $B.false$, которые назначаются первым командам кодов S_1 и S_2 соответственно для передачи им управления в зависимости от истинности или ложности B . В качестве последней команды кода S_1 формируется команда безусловного перехода **goto** $S.next$ для пропуска кода S_2 . После кода S_2 непосредственно следует команда с меткой $S.next$.

В операторе цикла **while** первая команда совпадает с первой командой кода для выражения B (рис. 9, в). В связанных с продукцией семантических правилах для передачи управления этой команде создается новая метка с использованием локальной переменной beg . Новая метка $B.true$ создается для назначения первой команде кода, формируемого для оператора S_1 , для передачи ей управления в случае истинности B . В конце кода для S_1 помещается команда **goto** beg для передачи управления в начало кода выражения B . Метка $B.false$ устанавливается равной $S.next$ для реализации перехода в случае ложности B к первой команде кода, следующего после оператора S . Метка $S_1.next$ устанавливается равной beg , чтобы при наличии в коде S_1 команд переходов управление передавалось в начало цикла к команде с меткой beg .

В операторе цикла **repeat** первая команда совпадает с первой командой кода для оператора S_1 (рис. 9, г). В связанных с продукцией семантических правилах для передачи управления этой команде создается новая метка $B.false$. Метка $B.true$ устанавливается равной $S.next$ для реализации перехода в случае истинности B к первой команде кода, следующего после оператора S .

Семантические правила для остальных продукций очевидны и не требуют дополнительных пояснений.

В общем случае реализация семантических правил может привести к тому, что одной трехадресной команде может быть назначено нескольких меток. Удаление лишних меток можно реализовать в процессе оптимизации кода. Другой подход основан на применении метода обратных поправок (подразд. 6.5), в котором метки создаются только тогда, когда они необходимы.

Если не определен синтаксический контекст особенностей использования логических и арифметических выражений, можно использовать для выражений атрибут, который позволит отличать типы выражений друг от друга и выбирать соответствующие способы генерации промежуточного кода. Подробнее об этом можно посмотреть в работе [2].