Решение задачи №5. Вариант 18

Условие: Построить промежуточный код для фрагмента исходного кода (F_4) в формате троек (C_3), выполнить оптимизацию и представить результат в формате трёхадресного кода (C_1).

1 Исходный фрагмент кода (F4)

В соответствии с вариантом 18 выберем фрагмент кода, содержащий цикл и возможности для оптимизации (например, вычисление общих подвыражений):

```
//                      F4
while (i < n) {
    a = b + c;
    d = b + c;
    i = i + 1;
}
```

2 Построение промежуточного кода в формате троек (C3)

Тройки представляют собой структуру $(, _1, _2)$. Ссылки на результат операции осуществляются по номеру строки (в скобках).

№	Оператор	Арг. 1	Арг. 2	Примечание
(0)	$<$	i	n	Сравнение для цикла
(1)	JNZ	(0)	(3)	Переход к телу цикла, если истина
(2)	JMP		(10)	Выход из цикла
(3)	$+$	b	c	Вычисление b + c (для a)
(4)	$=$	a	(3)	Присваивание a
(5)	$+$	b	c	Повторное вычисление b + c (для d)
(6)	$=$	d	(5)	Присваивание d
(7)	$+$	i	1	Инкремент i
(8)	$=$	i	(7)	Присваивание i
(9)	JMP		(0)	Переход на проверку условия
(10)	...			Конец

3 Типология команд промежуточного кода

Согласно теоретическим сведениям, команды промежуточного кода делятся на следующие типы:

- **Арифметические операции:** ($+$, $-$, $*$, $/$). Принимают два операнда и возвращают результат.
- **Операции отношения:** ($<$, $>$, $<=$, $>=$, $==$, $!=$). Используются для формирования логических условий.

- **Команды передачи управления:**
  - *Безусловный переход (JMP):* Переход на указанную метку или строку.
  - *Условный переход (JNZ/JZ):* Переход, если предыдущее условие истинно/ложно.
- **Операции присваивания:** (=). Запись вычисленного значения или константы в переменную.

4 Оптимизация промежуточного кода

Проведем оптимизацию методом **удаления общих подвыражений** (Common Subexpression Elimination) и **распространения копий**.

Протокол действий:

1. **Анализ:** Замечено, что выражение $b + c$ вычисляется дважды (строки (3) и (5)) без изменения переменных b и c между ними.
2. **Действие:** Заменим результат строки (5) ссылкой на результат строки (3).
3. **Результат:** Строка (5) становится избыточной. Присваивание d (строка 6) теперь использует результат строки (3).

5 Результирующий трёхадресный код (C1)

Преобразуем оптимизированные тройки в трёхадресный код (формат: $= {}_1$ Оператор ${}_2$).

Метка	**Команда трёхадресного кода**
L1:	$t_1 = i < n$
	if $t_1 == 0$ goto L2
	$t_2 = b + c$
	$a = t_2$
	$d = t_2$ // Оптимизировано (использовано t_2)
	$i = i + 1$
	goto L1
L2:	...

Вывод: В результате оптимизации количество арифметических операций внутри тела цикла сократилось, что повышает производительность итогового объектного кода.