МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«**Вятский государственный университет**»

**(ФГБОУ ВПО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

КОМПИЛЯТОР ЯЗЫКА PASCAL

Отчет

Лабораторная работа №5 по дисциплине

«Системное программное обеспечение»  
Вариант 19

Выполнил студент группы ИВТ-31 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Жукова М.Н./

Проверил преподаватель кафедры ЭВМ \_\_\_\_\_\_\_\_\_/Караваева О.В./

Киров 2016

1 Цель лабораторной работы

Целью данной лабораторной работы является закрепление знаний о процессе компиляции.

В ходе лабораторной работы необходимо выполнить все основные этапы компиляции:

* лексический анализ;
* синтаксический анализ;
* оптимизация;
* генерация кода.

При выполнении этапа оптимизации рассматриваются два метода оптимизации: свёртка, исключение лишних операций.

2 Исходный текст программы на языке Pascal:

procedure CCC;

var K,M:integer;

begin

K:=10;

M:=M+5;

end;

procedure DDD;

var K,B:integer;

begin

B:=10;

K:=B+6;

end;

var B,C:integer;

begin

B:=K+5;

if K>2 then B:=(K+5)-(K+4)

else CCC;

DDD;

end.

3Лексический анализ

Заполним внутренние таблицы компилятора:

Терминалы:

|  |  |  |
| --- | --- | --- |
| Символ | Индекс | Приоритет |
| BEGIN | 0 | 0 |
| + | 1 | 4 |
| - | 2 | 4 |
| \* | 3 | 5 |
| / | 4 | 5 |
| ( | 5 | 6 |
| ) | 6 | 3 |
| := | 7 | 2 |
| ; | 8 | 0 |
| END | 9 | 0 |
| END. | 10 | 0 |
| CALL | 11 | 1 |
| IF | 12 | 2 |
| THEN | 13 | 1 |
| ELSE | 14 | 0 |
| < | 15 | 3 |
| > | 16 | 3 |
| = | 17 | 3 |

Константы:

|  |  |
| --- | --- |
| Символ | Индекс |
| 10 | 1 |
| 5 | 2 |
| 6 | 3 |
| 2 | 4 |
| 4 | 5 |

Переменные:

|  |  |  |  |
| --- | --- | --- | --- |
| Символ | Индекс | Тип | Место |
| ССС | 1 | PROCNAME | 1 |
| K | 2 | INTEGER | 1 |
| M | 3 | INTEGER | 1 |
| DDD | 4 | PROCNAME | 2 |
| K | 5 | INTEGER | 2 |
| B | 6 | INTEGER | 2 |

4 Синтаксический анализ

Восходящий метод

С помощью восходящего метода строим таблицы триад и тетрад.

Таблица триад: Таблица тетрад:

** **

4.1Нисходящий метод

Дерево синтаксического разбора:

idProgram

idVariableSection:

idKeyWord: VAR

idIdent: B

idSymbol: :

idType: INTEGER

idSymbol: ;

idIdent: C

idSymbol: :

idType: INTEGER

idSymbol: ;

idOperationSection:

idKeyWord: BEGIN

idOperator:

idIdent: B

idSymbol: :=

idIdent: K

idSymbol: +

idConst: 5

idSymbol: ;

idOperator:

idSymbol: IF

idExpression:

idIdent: K

idSymbol: >

idConst: 2

idSymbol: THEN

idOperator:

idIdent: B

idSymbol: :=

idSymbol: (

idExpression:

idIdent: K

idSymbol: +

idConst: 5

idSymbol: )

idSymbol: -

idSymbol: (

idExpression:

idIdent: K

idSymbol: +

idConst: 4

idSymbol: )

idSymbol: ELSE

idOperator:

idSymbol: CALL

idIdent: CCC

idSymbol: ;

idOperator:

idSymbol: CALL

idIdent: DDD

idSymbol: ;

idKeyWord: END.

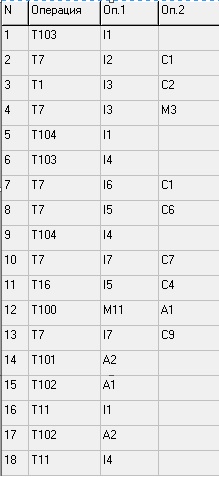
5 Оптимизация

После построения дерева синтаксического разбора выполняем этап оптимизации. На данном этапе выполняется исключение лишних операций за счет однократного программирования общих подвыражений и свёртка, т.е. выполнение операций, операнды которых известны во время компиляции.

5.1 Свёртка

Исключение лишних операции возможно при повторном вычислении выражения.

После свертки получаем таблицу:



5.2 Исключение лишних операций

Таблица после исключения лишних операций:



6 Генерация кода

После выполнения этапа оптимизации выполняем генерацию программы на языке Assembler.

Текст программы:

.MODEL small

.DATA

K\_1 dw ?

M\_1 dw ?

K\_2 dw ?

B\_2 dw ?

B dw ?

C dw ?

R1 dw ?

Const1 dw 10

Const2 dw 5

Const3 dw 6

Const4 dw 2

Const5 dw 4

Const6 dw 16

Const7 dw 21

Const8 dw 20

Const9 dw 1

.CODE

PROC CCC

MOV AX,Const1

MOV K\_1,AX

MOV AX,M\_1

ADD AX,Const2

MOV R1,AX

MOV AX,R1

MOV M\_1,AX

RET

ENDP CCC

PROC DDD

MOV AX,Const1

MOV B\_2,AX

MOV AX,Const6

MOV K\_2,AX

RET

ENDP DDD

Main:

MOV AX,Const7

MOV B,AX

MOV AX,Const4

CMP K\_2,AX

JNG A1

MOV AX,Const9

MOV B,AX

JMP A2

A1:

CALL CCC

A2:

CALL DDD

END Main

7 Построение дерева грамматического разбора восходящим методом

Необходимо вручную построить дерево синтаксического разбора восходящим методом.

Первым шагом при разработке процессора грамматического разбора, основанного на методе операторного предшествования, должно быть установление отношений предшествования между операторами грамматики. При этом под оператором понимается любой терминальный символ.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **if** | **then** | **else** | **(** | **)** | **:=** | **>** | **-** | **+** | **id** | **;** |
| **if** |  | **<·** |  |  |  |  |  |  |  | **<·** |  |
| **then** | **·>** |  | **<·** |  |  |  |  |  |  |  |  |
| **else** |  | **·>** |  |  |  |  |  |  |  |  |  |
| **(** |  |  |  |  | = |  |  | = | = |  |  |
| **)** |  |  |  |  |  |  |  |  |  |  |  |
| **:=** |  |  |  | = |  |  | **<·** |  |  | **<·** |  |
| **>** |  |  |  | = |  |  |  |  |  |  |  |
| **-** |  |  |  |  |  |  |  |  | = |  |  |
| **+** |  |  |  |  |  |  |  | = |  |  |  |
| **id** |  |  |  |  | **·>** | **=** |  |  |  |  |  |
| **;** | **·>** |  |  |  |  |  |  |  |  |  |  |

Дерево грамматического разбора приведено для самого длинного предложения

if K>2 then B:=(K+5)-(K+4)

Анализируем предложение по лексемам слева направо. Для каждой пары соседних операторов определено отношение предшествования.

