BÁO CÁO THỰC HÀNH CHƯƠNG TRÌNH DỊCH

BÀI 5: SINH MÃ CƠ BẢN

Họ và tên: Vũ Văn Minh

MSSV: 20225747

I. Tổng quan

1. Mục tiêu

   Xây dựng module **Code Generator** cho trình biên dịch ngôn ngữ KPL (tập con của Pascal), với nhiệm vụ chuyển đổi cây cú pháp thành tập lệnh thực thi trên máy ảo dựa trên ngăn xếp (Stack-based Virtual Machine).

2. Kiến trúc

   Gồm 4 module chính hoạt động tuần tự:

   - Scanner & Parser: Phân tích từ vựng và cú pháp.
   - Semantics & SymTab: Kiểm tra ngữ nghĩa và quản lý bảng ký hiệu (tầm vực, địa chỉ biến).
   - Code Generator: Sinh mã máy ảo.
     - Mô hình máy ảo: Sử dụng ngăn xếp (Stack) để tính toán và lưu trữ.
     - Tập lệnh: Bao gồm nhóm lệnh truy xuất bộ nhớ (Load/Store), tính toán số học/logic, điều khiển luồng (Jump) và quản lý chương trình con (Call/Return).

3. Thành phần

   Quản lý địa chỉ (codegen.c):

   - Xử lý truy xuất biến cục bộ và biến ngoài tầm vực (sử dụng Static Link).
   - Hỗ trợ truyền tham số theo tham trị (Value) và tham chiếu (Reference).

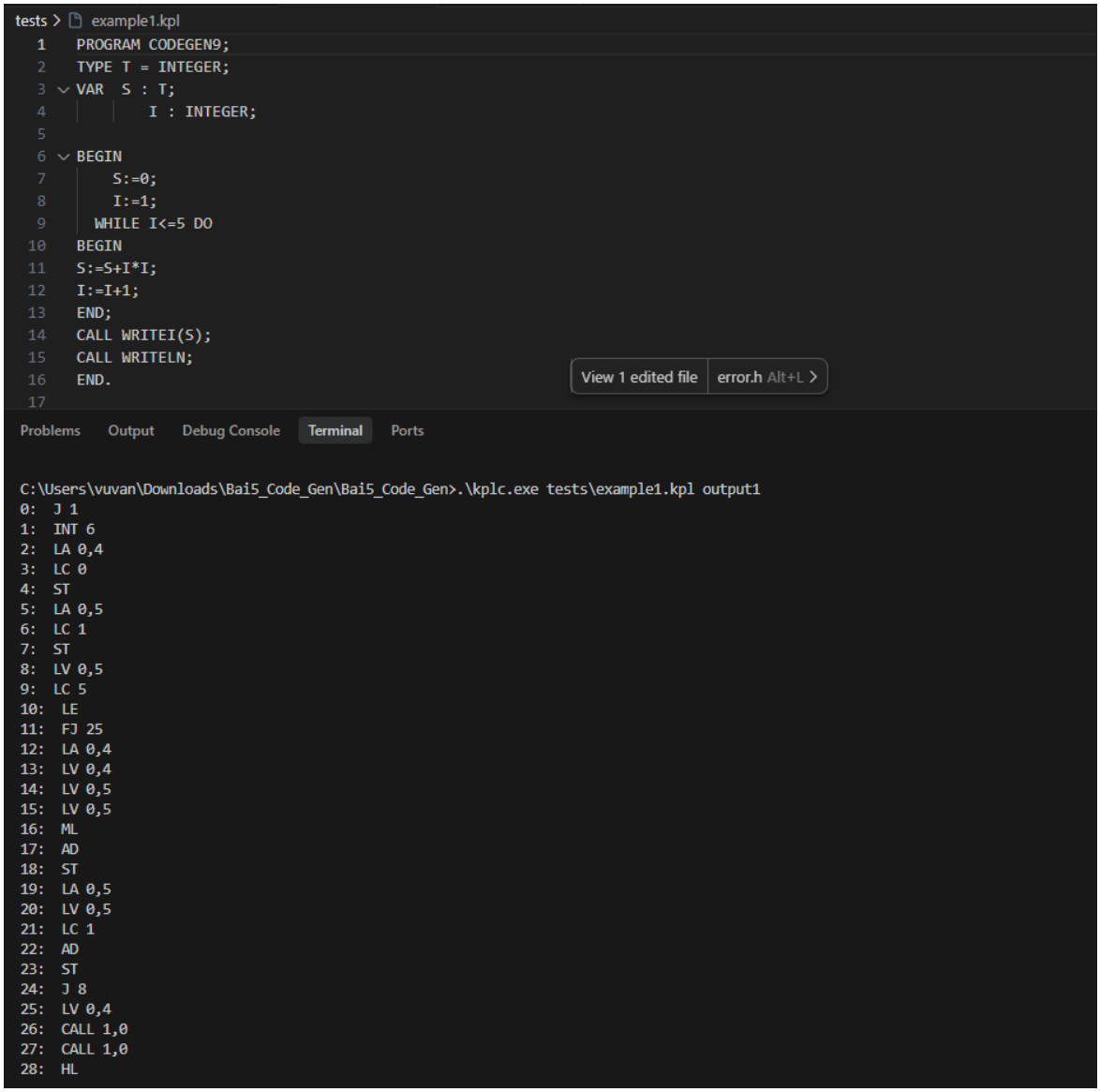
   Cấu trúc điều khiển (parser.c):

   - Sinh mã cho câu lệnh rẽ nhánh IF-THEN-ELSE và vòng lặp WHILE, FOR.
   - Sử dụng kỹ thuật Backpatching (quay lui cập nhật nhãn) để điều phối lệnh nhảy J và FJ .

   Biểu thức & Lời gọi hàm:

   - Sinh mã cho các phép toán số học, so sánh.
   - Điều khiển lời gọi hàm/thủ tục (CALL), bao gồm việc truyền ngữ cảnh (Scope) chính xác.

II. Kết quả thực hiện chương trình với 1 tệp không chứa chương trình con (Example1.kpl)



1. Phân tích mã nguồn

Mục đích: Tính tổng bình phương của các số nguyên từ 1 đến 5.

```
PROGRAM CODEGEN9;
TYPE T = INTEGER;
VAR  S : T;        (* Biến tích lũy tổng *)
    I : INTEGER;  (* Biến đếm vòng lặp *)

BEGIN
        S:=0;          (* Khởi tạo tổng bằng 0 *)
        I:=1;          (* Khởi tạo biến đếm bắt đầu từ 1 *)
  WHILE I<=5 DO   (* Lặp khi I vẫn nhỏ hơn hoặc bằng 5 *)
   BEGIN
   S:=S+I*I;     (* Cộng dồn bình phương của I vào S *)
```

```
    I:=I+1;       (* Tăng biến đếm *)
    END;
  CALL WRITEI(S); (* In kết quả S ra màn hình *)
  CALL WRITELN;   (* Xuống dòng *)
END.
```

2. Phân tích mã sinh ra (codegen**)**

Dựa trên code C đã cài đặt, trình biên dịch sẽ sinh ra các chỉ thị máy ảo tương ứng cho các phần chính:

Gán giá trị khởi tạo (S:=0):

- LA 0, 4: Tải địa chỉ biến S (giả sử offset là 4).
- LC 0: Tải hằng số 0.
- ST : Lưu giá trị 0 vào địa chỉ của S.

Điều kiện vòng lặp (WHILE I<=5):

- Tạo nhãn L1 (địa chỉ bắt đầu lặp).
- Tải giá trị I và 5.
- Thực hiện lệnh so sánh LE (Less or Equal).
- FJ L2: Nếu kết quả so sánh là Sai (False), nhảy đến nhãn L2 (cuối vòng lặp).

Thân vòng lặp:

- Tính toán S + I * I.
- Các lệnh LA , LV (Load Value) sẽ được dùng liên tục để lấy giá trị S, I.
- Lệnh MU (Nhân) và AD (Cộng) thực hiện tính toán.
- J L1: Nhảy quay lại đầu vòng lặp để kiểm tra điều kiện.

In kết quả (CALL WRITEI(S)):

- Đẩy giá trị S (55) vào ngăn xếp.
- Gọi thủ tục WRITEI cài sẵn.

III. Kết quả thực hiện chương trình với 1 tệp có chứa chương trình con (Example3.kpl)

1. Phân tích mã nguồn

```
PROGRAM  EXAMPLE3;  (* TOWER OF HANOI *)
VAR  I:INTEGER;
     N:INTEGER;
     P:INTEGER;
     Q:INTEGER;
     C:CHAR;

PROCEDURE  HANOI(N:INTEGER;  S:INTEGER;  Z:INTEGER);
BEGIN
  IF  N != 0  THEN
    BEGIN
      CALL  HANOI(N-1,S,6-S-Z);
      I:=I+1;
      CALL  WRITELN;
      CALL  WRITEI(I);
      CALL  WRITEI(N);
      CALL  WRITEI(S);
      CALL  WRITEI(Z);
      CALL  HANOI(N-1,6-S-Z,Z)
    END
END;  (*END OF HANOI*)

BEGIN
  FOR  N := 1  TO  4  DO
    BEGIN
      FOR  I:=1  TO  4  DO
        CALL  WRITEC(' ');
      C :=  READC;
      CALL  WRITEC(C)
    END;
  P:=1;
  Q:=2;
  FOR  N:=2  TO  4  DO
    BEGIN
      I:=0;
      CALL  HANOI(N,P,Q);
      CALL  WRITELN
    END
END.  (* TOWER OF HANOI *)
```

Chương trình bao gồm:

1. PROCEDURE HANOI: Là hàm đệ quy dùng để giải bài toán Tower of Hanoi.
    i. Hàm nhận **3 tham số**:
        1. N: số đĩa
        2. S: cột nguồn
        3. Z: cột đích
    ii. **Nguyên lý hoạt động**: Nếu N ≠ 0 thì:
        1. Gọi đệ quy để di chuyển N-1 đĩa.
        2. In ra thông tin bước di chuyển đĩa.
        3. Tiếp tục gọi đệ quy để hoàn tất quá trình.

2. Main Program

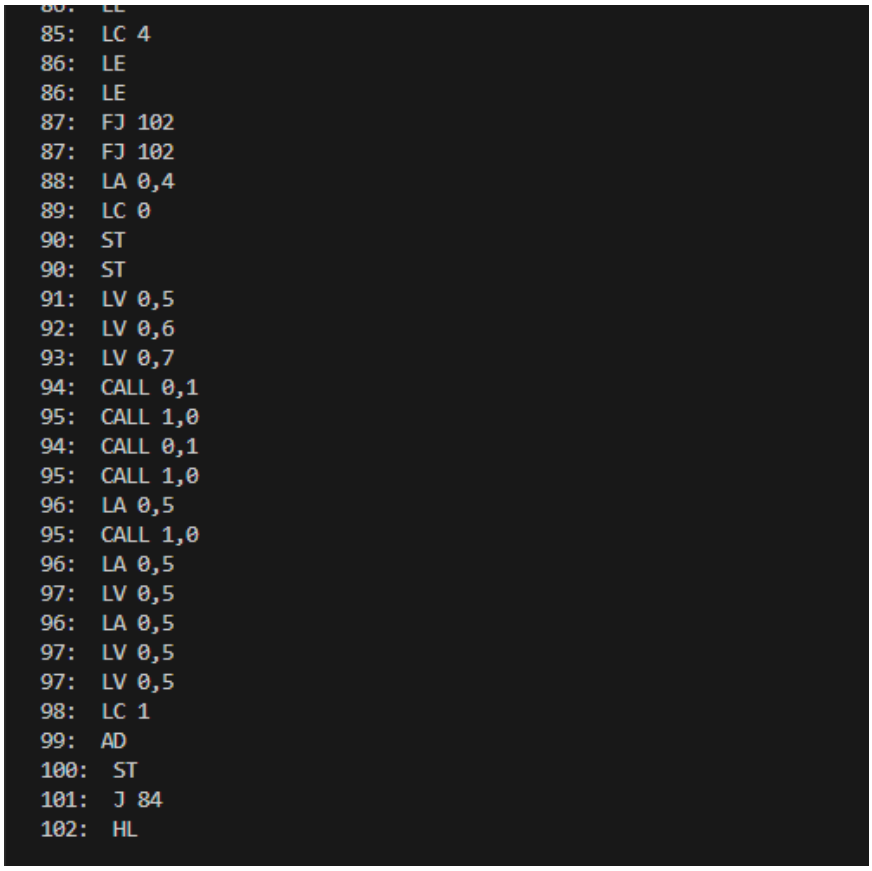
- Sử dụng **vòng lặp FOR** để:
    - Đọc **4 ký tự** từ bàn phím.
    - In các ký tự đó ra màn hình.

- Khởi tạo giá trị ban đầu:
  - P = 1
  - Q = 2
- Sử dụng **vòng lặp FOR**:Gọi thủ tục HANOI với số đĩa N lần lượt từ **2 đến 4**.

2. Phân tích mã sinh ra

```
C:\Users\vuvan\Downloads\Bai5_Code_Gen\Bai5_Code_Gen>.\kplc.exe tests\example3.kpl output3
0:  J 41
1:  J 2
2:  INT 7
3:  LV 0,4
4:  LC 0
5:  NE
6:  FJ 41
7:  LV 0,4
8:  LC 1
9:  SB
10:  LV 0,5
11:  LC 6
12:  LV 0,5
13:  SB
14:  LV 0,6
15:  SB
16:  CALL 1,1
17:  LA 1,4
18:  LV 1,4
19:  LC 1
20:  AD
21:  ST
22:  CALL 2,0
23:  LV 1,4
24:  CALL 2,0
25:  LV 0,4
26:  CALL 2,0
27:  LV 0,5
28:  CALL 2,0
29:  LV 0,6
30:  CALL 2,0
31:  LV 0,4
32:  LC 1
33:  SB
34:  LC 6
35:  LV 0,5
36:  SB
37:  LV 0,6
38:  SB
39:  LV 0,6
40:  CALL 1,1
41:  INT 9
42:  LA 0,5
43:  LC 1
44:  ST
45:  LV 0,5
46:  LC 4
47:  LE
48:  FJ 75
```

```
49:  LA 0,4
50:  LC 1
51:  ST
52:  LV 0,4
53:  LC 4
54:  LE
55:  FJ 64
56:  LC 0
57:  CALL 1,0
58:  LA 0,4
59:  LV 0,4
60:  LC 1
61:  AD
62:  ST
63:  J 52
64:  LA 0,8
65:  CALL 1,0
66:  ST
67:  LV 0,8
68:  CALL 1,0
69:  LA 0,5
70:  LV 0,5
71:  LC 1
72:  AD
73:  ST
74:  J 45
75:  LA 0,6
76:  LC 1
77:  ST
77:  ST
78:  LA 0,7
79:  LC 2
80:  ST
81:  LA 0,5
82:  LC 2
83:  ST
84:  LV 0,5
85:  LC 4
86:  LE
78:  LA 0,7
79:  LC 2
80:  ST
81:  LA 0,5
82:  LC 2
83:  ST
84:  LV 0,5
85:  LC 4
86:  LE
```



Khởi tạo và Quản lý Stack Frame (Thủ tục HANOI)

1. Khởi tạo Stack Frame cho thủ tục HANOI

Mã nguồn:

PROCEDURE HANOI(N: INTEGER; S: INTEGER; Z: INTEGER);

Mã máy:

2: INT 7 ; Cấp phát Stack Frame kích thước 7 words

Giải thích:

- Lệnh INT 7 dùng để cấp phát **Stack Frame** cho mỗi lần gọi thủ tục HANOI.
- Kích thước 7 words bao gồm:
  - 4 word hệ thống: *Static Link, Dynamic Link, Return Address, ...*
  - 3 word dành cho các tham số: N, S, Z
- Cách tổ chức này đảm bảo mỗi lần gọi đệ quy có **vùng nhớ độc lập**, không ảnh hưởng lẫn nhau.

2. Kiểm tra điều kiện dừng đệ quy (Base Case)

Mã nguồn:

IF N != 0 THEN

Mã máy:

3: LV 0, 4 ; Load giá trị tham số N

4: LC 0 ; Load hằng số 0

5: NE ; So sánh N != 0

6: FJ 40 ; Nếu N == 0 thì nhảy ra khỏi thủ tục

Giải thích:

- LV 0,4 truy xuất tham số N trong Stack Frame hiện tại.
- Lệnh NE kiểm tra điều kiện $N \neq 0$.
- FJ 40 (False Jump) thực hiện nhảy có điều kiện khi biểu thức sai ($N = 0$), đảm bảo kết thúc đệ quy đúng với điều kiện dừng.

3. Chuẩn bị tham số và Gọi đệ quy

Mã nguồn:

CALL HANOI(N-1, S, 6-S-Z);

Mã máy:

7: LV 0, 4 ; Tải N
8: LC 1
9: SB ; N - 1
10: LV 0, 5 ; Tải S
11: LC 6
12: LV 0, 5
13: SB ; 6 - S
14: LV 0, 6 ; Tải Z
15: SB ; (6 - S) - Z
16: CALL 0, 2 ; Gọi thủ tục HANOI

Giải thích:

- Các tham số được **tính toán và đưa vào stack** theo đúng thứ tự trước khi gọi thủ tục.
- Lệnh CALL 0,2:
  - 0 là **Level Difference**, cho biết lời gọi trong cùng phạm vi (scope).
  - 2 là địa chỉ bắt đầu của thủ tục HANOI.
- Khi gọi, máy ảo tạo một **Stack Frame mới** cho lời gọi đệ quy.

4. Truy xuất biến toàn cục (Global Variable Access)

Mã nguồn:

I := I + 1;

Mã máy:

17: LA 1, 4 ; Lấy địa chỉ biến I (scope cha)
18: LV 1, 4 ; Lấy giá trị I
19: LC 1
20: AD ; I + 1
21: ST ; Gán lại cho I

Giải thích:

- Tham số 1 trong các lệnh LA và LV cho biết biến I **không nằm trong Stack Frame hiện tại**, mà thuộc **scope cha** (chương trình chính).

- Điều này chứng tỏ cơ chế **Static Link** đã được xử lý chính xác trong quá trình sinh mã.

5. Vòng lặp trong chương trình chính (Main Loop)

Mã nguồn:

```
FOR N := 1 TO 4 DO
```

Mã máy:

```
42: LA 0, 5   ; Địa chỉ biến N
43: LC 1
44: ST        ; N := 1
45: LV 0, 5
46: LC 4
47: LE        ; N <= 4
48: FJ 78     ; Thoát vòng lặp nếu sai
...
76: J 45      ; Quay lại kiểm tra điều kiện
```

Giải thích:

- Mã máy thể hiện đúng cấu trúc vòng lặp FOR:
    1. Khởi tạo biến đếm
    2. Kiểm tra điều kiện
    3. Thực hiện thân vòng lặp
    4. Tăng biến đếm
    5. Quay lại kiểm tra điều kiện
- Cách sinh mã này đảm bảo **ngữ nghĩa chuẩn** của vòng lặp FOR.