BÁO CÁO SINH MÃ

Họ và tên: Lê Nguyễn Minh Đức

MSSV: 20225611

PHẦN I: KIẾN TRÚC

1.1 MÔ HÌNH STACK-BASED

Máy sử dụng mô hình stack (not register-based):

- Tất cả phép toán thông qua stack

- Biến lưu trên stack frame

- Procedure frames được push/pop khi gọi

Stack Layout:

← SP (Stack Pointer)

| Local vars |

| Parameters |

| Static link | (offset 3)

| Return addr | (offset 2)

| Dynamic link | (offset 1)

| Return val | (offset 0)

← FP (Frame Base)

1.2 CÁC LỆNH MÁY ẢO

a) Load Instructions (đẩy giá trị lên stack)

LA level,offset Load Address - đẩy địa chỉ

LV level,offset Load Value - đẩy giá trị

LC constant Load Constant - đẩy hằng số

LI Load Indirect - load từ địa chỉ trên stack

b) Store Instruction

ST Store - lưu stack top vào địa chỉ

c) Arithmetic Instructions

AD Addition (cộng)

SB Subtraction (trừ)

ML Multiplication (nhân)

DV Division (chia)

NEG Negation (đảo dấu)

d) Comparison Instructions

EQ Equal (bằng)

NE Not equal (khác)

LT Less than (nhỏ hơn)

LE Less or equal (nhỏ hơn hoặc bằng)

GT Greater than (lớn hơn)

GE Greater or equal (lớn hơn hoặc bằng)

e) Control Flow Instructions

J label Jump (nhảy vô điều kiện)

FJ label False Jump (nhảy nếu top = 0)

f) Procedure Instructions

CALL level,addr Call procedure/function

EP End Procedure

EF End Function

HL Halt

g) I/O Instructions (placeholder)

RI Read Integer

RC Read Character

WRI Write Integer

WRC Write Character

WLN Write Line

h) Frame Instructions

INT delta Initialize frame (allocate stack)

DCT delta Decrement top of stack

PHẦN II: QUÁ TRÌNH SINH MÃ

2.1 ASSIGNMENT STATEMENT (GÁN GIÁ TRỊ)

Loại 1: Gán hằng số

Code KPL: S := 5

Sinh mã: LA 0,4 (Load address of S at offset 4)

LC 5 (Load constant 5)

ST (Store)

Stack flow:

Before: []

LA: [address_of_S]

LC: [address_of_S, 5]

ST: [] (5 stored at address_of_S)

Result: S = 5

Loại 2: Gán biến khác

Code KPL: X := Y

Sinh mã: LA 0,X_offset

LV 0,Y_offset

ST

Loại 3: Gán từ hàm/procedure

Code KPL: X := READI()

Sinh mã: LA 0,X_offset

RI (Read integer from input)

ST

2.2 ARITHMETIC EXPRESSIONS (BIỂU THỨC SỐ HỌC)

Ví dụ 1: Phép cộng

Code KPL: Z := A + B

Sinh mã: LA 0,Z_offset

LV 0,A_offset

LV 0,B_offset

AD (Pop B, Pop A, Push A+B)

ST

Stack evolution:

LA: [addr_Z]

LV A: [addr_Z, A]

LV B: [addr_Z, A, B]

AD: [addr_Z, A+B]

ST: []

Ví dụ 2: Phép nhân kết hợp cộng

Code KPL: S := S + I * I

Sinh mã: LA 0,S_offset

LV 0,S_offset (Load S)

LV 0,I_offset (Load I, first operand for multiply)

LV 0,I_offset (Load I, second operand for multiply)

ML (I*I)

AD (S + I*I)

ST

Stack evolution:

LA: [addr_S]

LV S: [addr_S, S]

LV I: [addr_S, S, I]

LV I: [addr_S, S, I, I]

ML: [addr_S, S, I*I]

AD: [addr_S, S+I*I]

ST: []

Ví dụ 3: Biểu thức phức tạp

Code KPL: X := (A + B) * (C - D)

Sinh mã: LA 0,X_offset

LV 0,A_offset

LV 0,B_offset

AD ((A+B))

LV 0,C_offset

LV 0,D_offset

SB ((C-D))

ML ((A+B) * (C-D))

ST

2.3 (SO SÁNH & ĐIỀU KIỆN)

Ví dụ: IF N != 0 THEN ... ELSE ... END

Sinh mã: LV 0,N_offset (Load N)

LC 0 (Load 0)

NE (Compare N != 0, push true/false)

FJ else_label (Jump to else if false)

[THEN code]

J end_label (Jump over else)

else: [ELSE code]

end: [Continue]

Stack evolution:

LV N: [N]

LC 0: [N, 0]

NE: [result] (1 if N!=0, 0 if N==0)

FJ: [] (Pop result, conditional jump)

Loại so sánh được hỗ trợ:

EQ, NE, LT, LE, GT, GE

2.4 LOOPS - WHILE

Code KPL: WHILE I <= 5 DO

S := S + 1

END

Sinh mã: [start] LV 0,I_offset (Check condition)

LC 5

LE

FJ end_label (Exit if false)

[body] LA 0,S_offset (Loop body)

LV 0,S_offset

LC 1

AD

ST

J start_label (Jump back to condition)

[end] ... (Continue after loop)

Điểm quan trọng:

- Condition được check ở đầu mỗi iteration
- FJ jump tới end nếu false
- J jump trở lại start để check lại
- Giải phóng stack trước khi jump

2.5 LOOPS - FOR

Code KPL: FOR I := 1 TO 5 DO

X := I * I

END

Sinh mã: LA 0,I_offset (Initialize)

LC 1

ST

[check] LV 0,I_offset (Check I <= 5)

LC 5

LE

FJ end_label

[body] LA 0,X_offset (Loop body: X := I*I)

LV 0,I_offset

LV 0,I_offset

```
        ML
        ST
    [incr]  LA 0,I_offset    (Increment I)
        LV 0,I_offset
        LC 1
        AD
        ST
        J check_label    (Jump back to check)
    [end]   ...
```

2.6 PROCEDURE CALLS

Ví dụ 1: Call procedure không tham số

Code KPL: CALL WRITELN

Sinh mã: WLN (hoặc HL, tạm thời)

Ví dụ 2: Call procedure có tham số

Code KPL: CALL WRITEI(S)

Sinh mã: LV 0,S_offset (Push argument)

WRI (Write integer)

Ví dụ 3: Call user-defined procedure

Code KPL: CALL HANOI(N-1, S, 6-S-Z)

Sinh mã: LV 0,N_offset (Argument 1: N)

```
        LC 1
        SB          (N-1)
        LV 0,S_offset   (Argument 2: S)
        LV 0,S_offset   (Argument 3: 6-S)
        LC 6
        SB
        LV 0,Z_offset
        SB          (6-S-Z)
        CALL level,addr (Call with 3 parameters)
```

2.7 PROCEDURE DEFINITIONS

Ví dụ: PROCEDURE TEST(N: INTEGER) BEGIN ... END

Sinh mã: J main_code_addr (Jump qua procedure definition)

[proc] INT frame_size (Allocate frame for procedure)

... procedure body code ...

EP (End procedure)

[main] ... main code ...

Cơ chế:

1. Tạo code address cho procedure (address 1)

2. Sinh J skip định nghĩa procedure

3. Trong procedure body:

- Sinh INT với frameSize = RESERVED_WORDS + paramCount + localVarSize
- Sinh code cho procedure body
- Sinh EP

4. Update skip jump tới main code

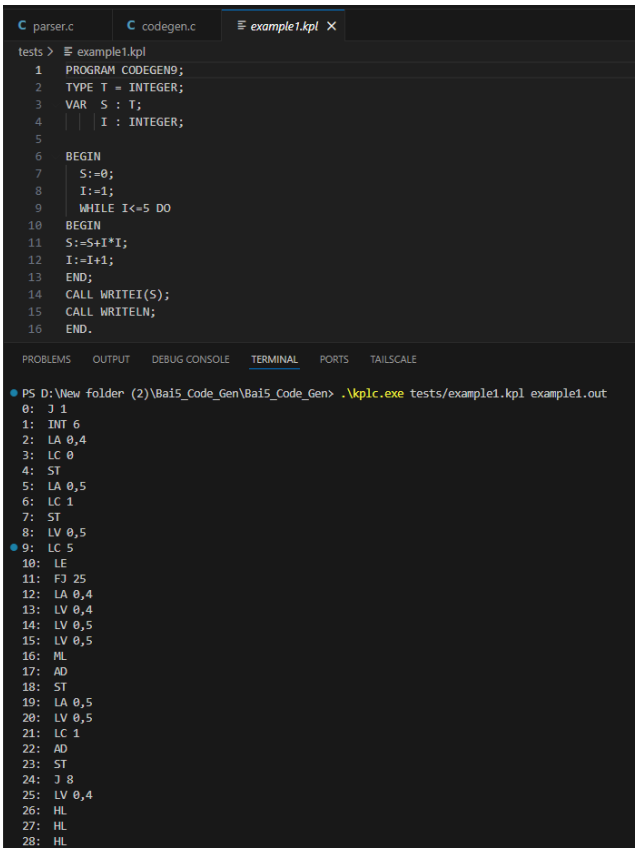
Recursive calls:

- CALL tạo frame mới trên stack
- Trả về từ EP/EF
- Dynamic link theo dõi caller frame

PHẦN III: TEST CASE 1

3.1 NGỮ CẢNH

File: tests/example1.kpl



3.2 EXECUTION TRACE

Khởi tạo:

Frame: [?, ?, ?, ?, ?, ?] (6 words uninitialized)

PC: 0

Lệnh 0-1:

PC=0: J 1 → Jump to PC=1

PC=1: INT 6 → Allocate 6 words

Frame: [0, 0, 0, 0, 0, 0]

Lệnh 2-7 (S := 0; I := 1):

PC=2-4: LA 0,4; LC 0; ST

Frame: [0, 0, 0, 0, 0, 0]

S := 0 (frame[4] = 0)

PC=5-7: LA 0,5; LC 1; ST

Frame: [0, 0, 0, 0, 0, 1]

I := 1 (frame[5] = 1)

Vòng lặp lần 1 (I=1):

PC=8-10: LV 0,5; LC 5; LE

Stack: [1] (1 <= 5 = true)

PC=11: FJ 25 → không jump (true)

PC=12-18: S := S + I*I

S = 0 + 1*1 = 1

PC=19-23: I := I + 1

I = 1 + 1 = 2

PC=24: J 8 → Jump back

Vòng lặp lần 2 (I=2):

Condition: 2 <= 5 = true

S := S + I*I = 1 + 4 = 5

I := I + 1 = 3

Vòng lặp lần 3 (I=3):

Condition: 3 <= 5 = true

S := S + I*I = 5 + 9 = 14

I := I + 1 = 4

Vòng lặp lần 4 (I=4):

Condition: 4 <= 5 = true

S := S + I*I = 14 + 16 = 30

I := I + 1 = 5

Vòng lặp lần 5 (I=5):

Condition: 5 <= 5 = true

S := S + I*I = 30 + 25 = 55

I := I + 1 = 6

Vòng lặp lần 6 (I=6):

PC=8-10: LV 0,5; LC 5; LE

Stack: [0] (6 <= 5 = false)

PC=11: FJ 25 → Jump to 25 (exit loop)

Sau loop (PC=25):

PC=25: LV 0,4 → Push S (= 55)

PC=26: HL → WRITEI (tạm thời)

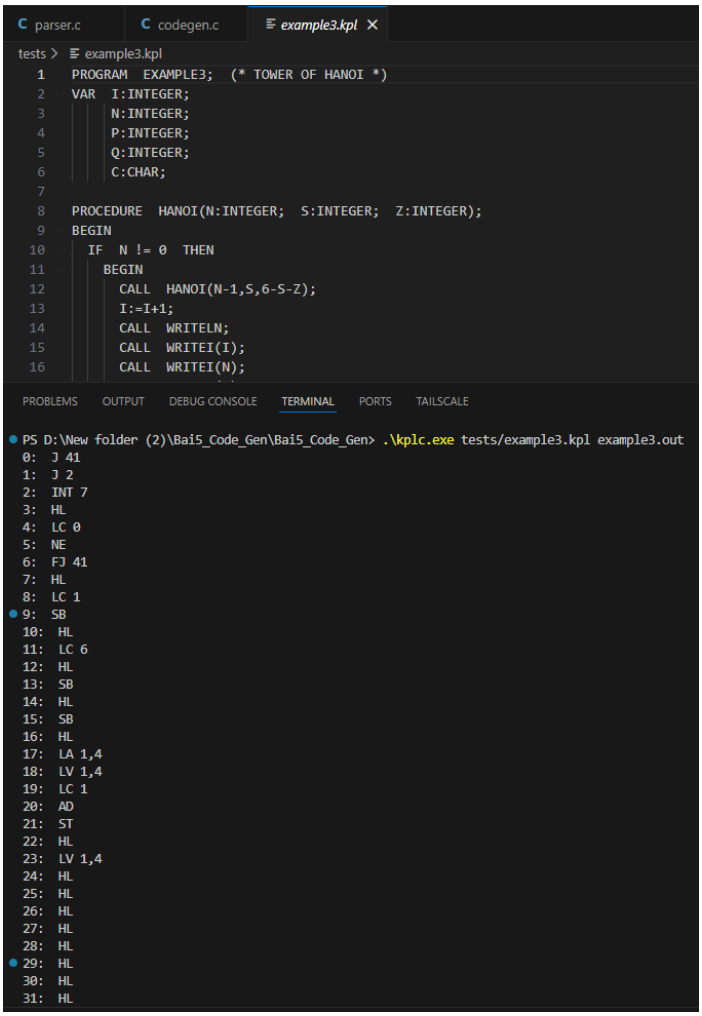
PC=27: HL → WRITELN (tạm thời)

PC=28: HL → Halt

PHẦN IV: TEST CASE 2

7.1 NGỮ CẢNH

File: tests/example3.kpl



7.4 RECURSIVE CALL ANALYSIS

CALL HANOI(4, 1, 2):

Stack growth:

Initial: [main frame]

|

CALL H(3,1,3): [H(3,1,3)]

[main frame]

|

CALL H(2,1,2): [H(2,1,2)]

[H(3,1,3)]

[main frame]

|

CALL H(1,1,3): [H(1,1,3)]

[H(2,1,2)]

[H(3,1,3)]

[main frame]

|

CALL H(0,1,2): [H(0,1,2)] ← Base case (N=0, không call)

[H(1,1,3)]

[H(2,1,2)]

[H(3,1,3)]

[main frame]

Mỗi frame có:

- Return address
- Dynamic link (caller frame)
- Parameters (N, S, Z)
- Return value

EP instruction:

- Restore FP từ dynamic link
- Jump to return address
- Back to caller