Mini Project 1

Goal

- 大致了解compiler的架構
- 了解從高階語言至組合語言的過程
- Trace code能力與實作能力

Compiler架構

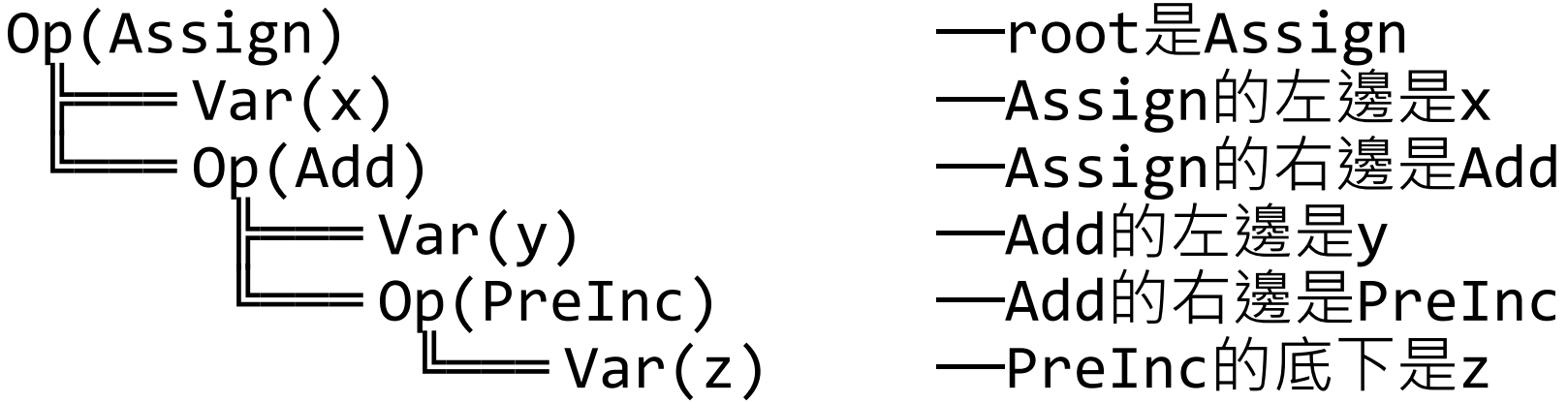
Lexer(Lexical Analyzer)

- 把輸入的東西切成一個一個的token
  - Input: x=y+++z
  - Token: Var(x), Op(Assign), Var(y), Op(Add), Op(PreInc), Var(z)

Compiler架構

Parser

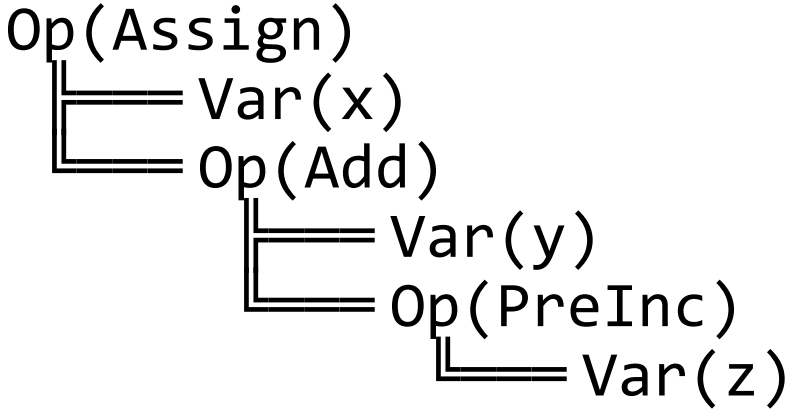
- 根據語法，利用Token建出正確的AST(Abstract Syntax Tree)
  - Token: Var(x), Op(Assign), Var(y), Op(Add), Op(PreInc), Var(z)
  - Tree:



Compiler架構

CodeGen

- 利用建立出的AST，把ASM生出來
  - Tree:



  - ASM(not real ASM):
    - load “y” into r0
    - store z+1 into r1
    - store r1 into “z”
    - add r0 and r1 into r0
    - store r0 into “x”

Compiler架構

其他:

1. 優化
   - Input: x=(1+3*6+8)-(5/2)
   - ASM(無優化): (一大坨，這裡不寫)
   - ASM(優化): `store 25 into “x”`
2. Semantic Check
   - 確認生成的AST是否有語法錯誤
     - `x=1+3` :OK!
     - `y=+-+-+-+-+-x` :OK!
     - `z=+++x` :Wrong!
     - `++(x++)` :Wrong!
     - `z=y=x+1+2+3+4*5` :OK!

Trace Code

struct TOKEN:

- 將輸入的expression變成一個個Token
- 將Token串成link list

內容：

int kind // 代表是哪一種物件 ex: 正號、加號、數字.....

int param // 數字的值、括號的編號、哪一個變數(x,y,z)

struct _TOKEN *prev, *next // link list 的prev和next

Trace Code

struct _AST:
。將Token建成二元樹後的節點

內容：
int type; // 代表是哪一種物件 ex: 正號、加號、數字.....
int val; // 數字的值、變數的值
struct _AST *lhs, *rhs, *mid; // 往左節點、右節點和中節點

Trace Code

AST *parser:
有幫大家做好的部分....

- 建立當前的節點：int mid = find_Tmid(arr, l, r)
- 括號的節點：if(r == findParPair(arr, l, r))
  - "a++"的"++"節點: if(getOpLevel(arr[mid].kind) == 1)
    — 所以是遞迴 l 到 mid-1

Trace Code

AST *parser:
沒幫大家做好的部分....

- ++a ?
- +-*/%= ?

Trace Code

void semantic_check:
有幫大家做好的部分....

。++、--、正號、負號 的偵錯
。括號的偵錯

Trace Code

void semantic_check:
沒幫大家做好的部分....

。+-*/%=？
。其他怪怪的錯？？？

Trace Code

void codegen(AST *ast)

。印出assmbly code
。做assembly code的優化？？？

Trace Code

直接來看看code的樣子吧~~~

assembly

- 包含多種instruction
  - `load`　把memory的資料存進register中
  - `store`　把register的資料存回memory中
  - `add`　加法運算
  - `sub`　減法運算
  - `mul`　乘法運算
  - `div`　除法運算
  - `rem`　取模運算(%)

assembly

- load
  - load reg [Addr]，花費200 cycle
  - Load r1 [4]: 把[4]的值存進r1中，也就是r1=y。
    - 若[4]=18, r1=10，load執行完後r1=[4]=18。
- Store
  - Store [Addr] reg，花費200 cycle
  - store [8] r3: 把r3的值存進[8]中，也就是[8]=r3。
    - 若[8]=-3, r3=102，store執行完後[8]=r3=102。

assembly

- add
  - add rd rs1 rs2，花費10 cycle
  - add r3 r6 r3: 把r6與r3的值相加後存在r3中，也就是r3=r6+r3。
    - 若r3=87, r6=1450，add執行完後r3=1537, r6=1450。
- 其他的(sub, mul, div, rem)也類似。
- sub: 10 cycle, mul: 30 cycle
- div: 50 cycle, rem: 60 cycle

評分

- 占總成績 8%
- 24個testcases，其中有6個範例testcase
- 總cycle數越少越好，班排前10%會拿到總成績2分的bonus

Submission / Demo

- 11/06(三) 晚上11:59前繳交code
  - 上傳到iLMS並命名為 學號.c
- Demo: 11/07(四) 晚上6:30起
  - 務必出席，否則以0分計算。若有事無法前來請先告知協調。
  - demo完成後會在一定時間內公布通過的測資數與總耗費的cycle數。

Questions?