MEAZZA COMPILER

蔡万鑫

ANTLR

- 用ANTLR做词法分析的好处
- ANTLR的新特性 支持左递归
- 优化后的左递归 VS 递归下降

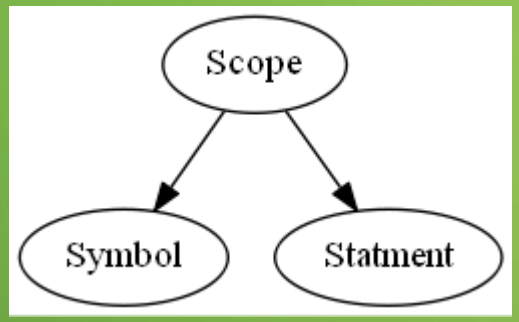
SYNTAX CHECK

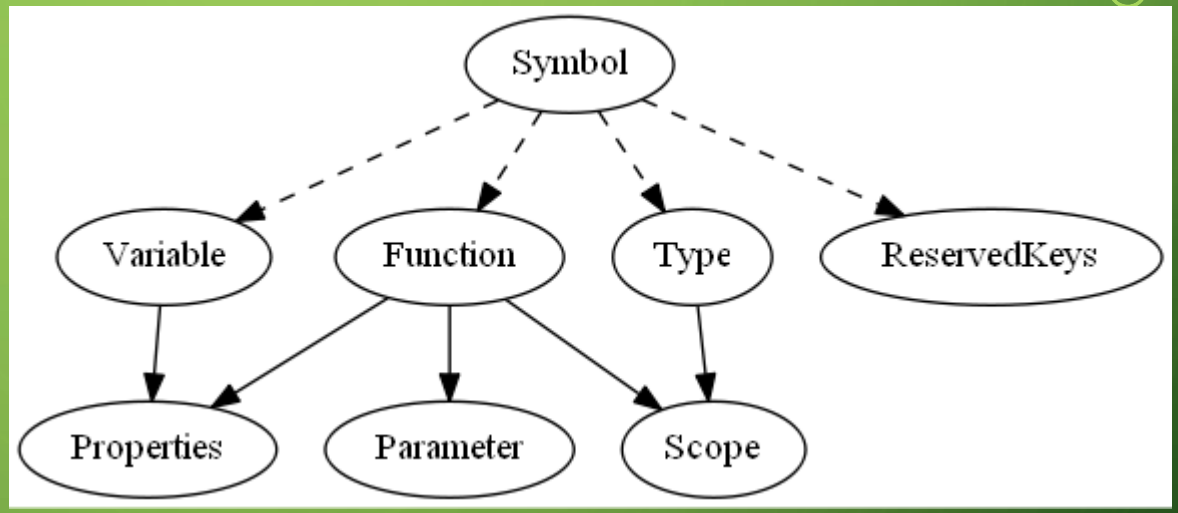
- 什么时候检查
- ANTLR的Visitor模式

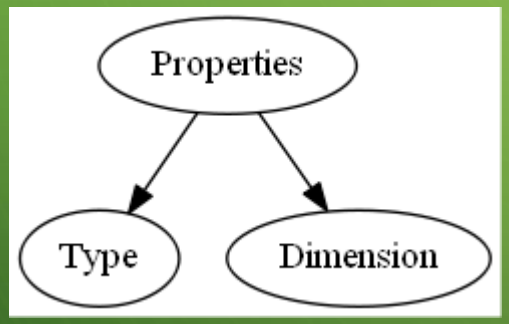
AST

- CST 转 AST
- Without Symbol Table

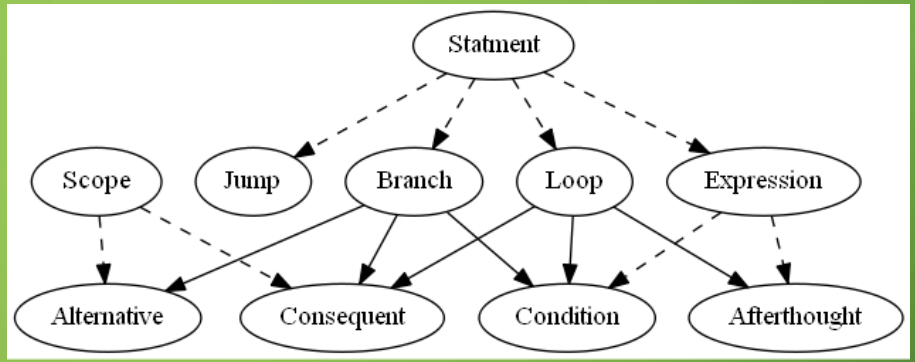
AST

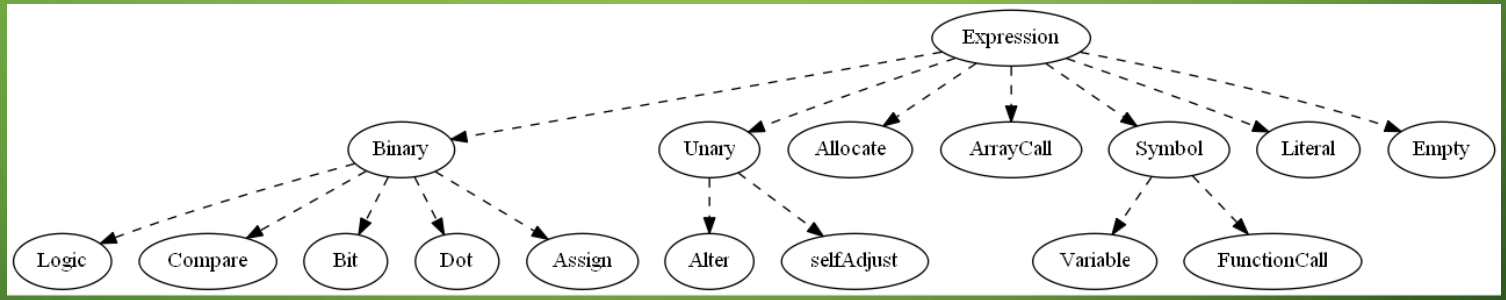

Scope
Symbol
Statment


Symbol
Variable
Function
Type
ReservedKeys
Properties
Parameter
Scope


Properties
Type
Dimension

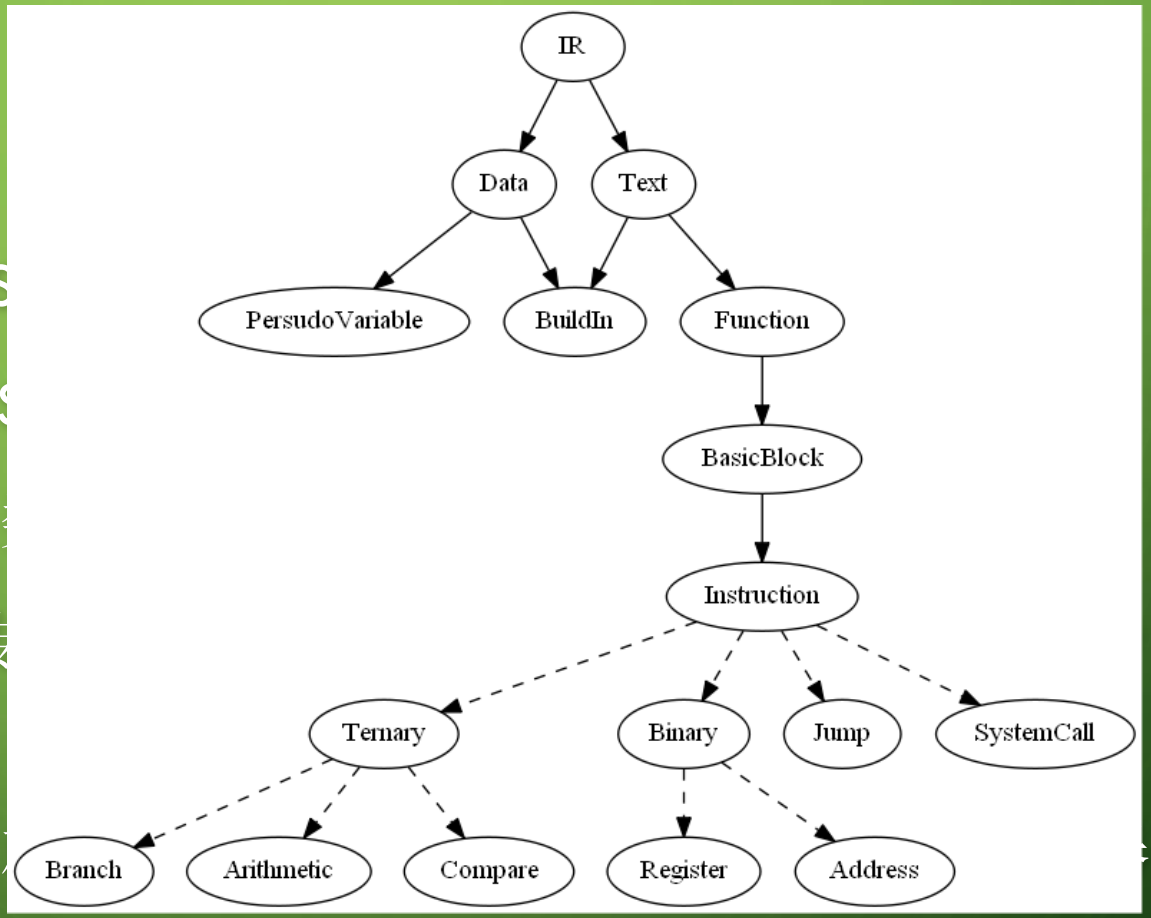
AST


Statment
Scope
Jump
Branch
Loop
Expression
Alternative
Consequent
Condition
Afterthought


Expression
Binary
Unary
Allocate
ArrayCall
Symbol
Literal
Empty
Logic
Compare
Bit
Dot
Assign
Alter
selfAdjust
Variable
FunctionCall

伪MIPS

- 思想： MIPS
- 对Symbol做S
- 全局变量、
- $a_0 \ldots a_3, t_0$传
  头指针
- 进入函数时



冗余代码删除

- Compare和Branch结合
- ~~如果没有RAW，则删去该语句~~
- 如果一个寄存器没有被Read，则删去Write语句
- ~~自己move到自己（可在分配完寄存器后实施）~~
- ~~多余move~~

MIPS寄存器分配

- Classify the virtual register : temporary / local / global / save in address

  (*在代码中的分类是 local , localSave , global, saveInAddress)

- ~~Optimize the usage~~

- Allocate & Deal with some cases

怎么CLASSIFY

- Temporary： 只在一个Basic Block 里面出现
- Local 在多个Basic Block 里面出现，但RAW不会跨越function call
- Global RAW跨越function call，但不会跨越可能导致自调用的call
- Save In Address RAW跨越自调用call
  - Frame size 就是 Save In Address的size

寄存器初始分配方案

- Temporary： $v_0, v_1, a_0 \dots a_4, t_0, r_a$
- Local： $t_1 \dots t_7$
- Global： $s_0 \dots s_6$
- Reserved：
  $s_7, (global\ variable)\ s_8(f_p)(global\ base\ adress)$
- Kernel： $\$0, \$1, \$g_p, \$k_1, \$k_2$

寄存器初始分配方案

- 先对于每个函数，将读写最多的local虚拟寄存器分配到$t_1 \dots t_7$
- 在这个函数里面$t_1 \dots t_7$唯一表示一个虚拟寄存器。

- 然后统计全局有global标记的虚拟寄存器，将读写最多的分配到$s_0 \dots s_6$
- $s_0 \dots s_6$在全局唯一表示一个虚拟寄存器。

寄存器分配优化

- 用f[i][k]表示第i个函数读写次数最多k个的local虚拟寄存器读写次数之和，g[k]表示读写次数最多的global虚拟寄存器读写次数之和

- Find k to maximize $\sum_i f[i][k] + g(14-k)$

寄存器分配优化

- 全局变量可以一开始不用放入内存，而是分配一个虚拟寄存器然后标记为global，如果没有分配到global寄存器再放入内存。

TEMPORARY寄存器的分配

- 在进入每个函数的时候check如果有没有用到的local和global寄存器，将它标记为temporary
- 将temporary标记的虚拟寄存器与temporary寄存器挂钩
- 如果虚拟寄存器生命周期结束，就释放寄存器
- 如果temporary寄存器都是占用的，将最早使用的temporary寄存器取出，在.data的VReg区域找一个空位放进去，然后腾出寄存器。

注意点

- 给Parameter传值的时候，应该给parameter所用的temporary寄存器加个lock。
- 压栈操作在寄存器分配后进行，如果发现一个函数的save in address是0就不去压栈。