ZJU C COMPILER

ZCC

Xiuye Gu, Haolin Fu, Qimai Li, Qingcheng Xiong

WHAT DOES ZCC SUPPORT?

ERROR HANDLING
AND ERROR RECOVERY

- Low level errors
  - Find them when building the Concrete Parsing Tree and recover most of them.
- High level errors
  - Find them when transferring the Concrete Parsing Tree to Abstract Parsing Tree.

LOW LEVEL ERRORS

WHAT CAN WE HANDLE AND RECOVERY

- 1. Missing semicolon
- 2. Missing right curly bracket
- 3. wrong identifiers (not conform to the identifier naming rule in C)
- 4. wrong characters after operators
  - ZCC can do error recovery for the above 4 rules and get the correct parsing tree.
- 5. Various statements that do not conform to ANSI C grammar. (error_pos.c)

RECOVER LOW LEVEL ERRORS

PRINCIPALS

- Adding error rules to our BNF
- Adding EOF token to handle the last missing right curly bracket
- Remove the error token from the parsing tree
- Insert the missing token into the parsing tree
- Using some counter to balance the curly brackets
- So that we can discover most common mistakes and do error recovery (build the correct parsing tree)

HIGH LEVEL ERRORS

WHEN CONCRETE TREE -> ABSTRACT TREE

- 1. When function declaration does not conform to its definition.
- 2. Repeated definition for variables.
- 3. Types not match when assigning value.
- 4. In expressions, the type of operand is not allowed in the grammar rules.
- 5. Typo.
  - Using edit distance to give the hint.
- 6. When calling a certain function, the parameter table does not confirm to its definition.
- 7. The return value of a function does not confirm to its definition.

BASIC X86 SUPPORT

- Calculation: add, sub, mul, div.
- Logic: and, or, not.
- Jump: jmp, je, jg, jl.
- Shift: sal, sar.
- Function: call, ret.
- Stack: push, pop.
- Float number operation: fld, fstp, fadd, fsub, fmul, fdiv.
- Global/Static variables, Constant float number, String

OPTIMIZATIONS

- Front-end optimization：
- constant folding
- dead code elimination
- Back-end optimization：
- Register optimization：
- set ebx,ecx,edx to be the temporary residential area for temp variables.
- set esi edi to be the register swap space for eax
- Command optimization：
- *2 / 4 / 8… ->sal (change multiplying two’s multiples to shift operation)
- lea 2*eax+offset -> reg