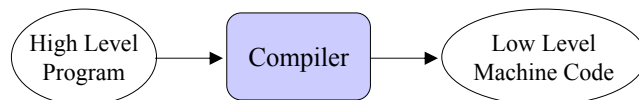


CS6353 Compiler Construction

What is a Compiler?

❖ A language translator



Easy to understand
User-friendly syntax
Machine-independent
e.g., Java, C

Hard to understand
Hardware specific
Involving registers &
memory locations

Related Concepts

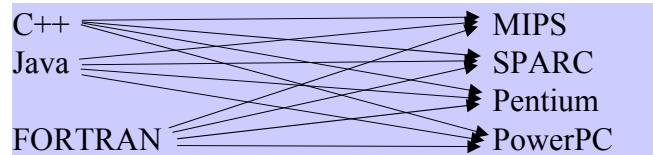
- ❖ **Compiler**
 - ❑ Translate from high-level source code to machine language or intermediate code
 - ❑ E.g., Pascal, C
- ❖ **Interpreter**
 - ❑ Translate the high-level program step by step and immediately execute the resulting instructions
 - ❑ E.g. Basic, Lisp
- ❖ **Interpreter versus compiler**
 - ❑ Interpreter is easier to implement, easier to modify, and easier to debug
 - ❑ But interpreter is less efficient and more space consuming

Related Concepts

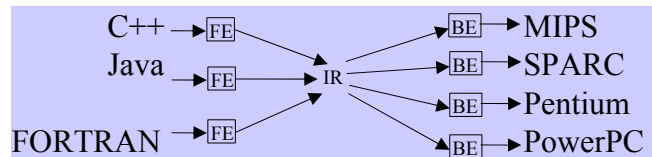
- ❖ **Two stages**
 - ❑ First compile the high-level program into intermediate code
 - ❑ Interpret the intermediate code during execution
 - ❑ E.g. Java
- ❖ **Assembler**
 - ❑ Translate assembly language to machine code
 - ❑ One to one correspondence
- ❖ **Compiler compiler**
 - ❑ Generate compiler based on language specifications

Related Concepts

❖ m source languages, n platforms $\Rightarrow m \cdot n$ translators



❖ Translate source code into IR (intermediate representation) then translate IR to machine code $\Rightarrow m+n$ translators

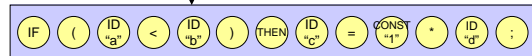


Phases of Modern Compilers

❖ Lexical analyzer (scanner)

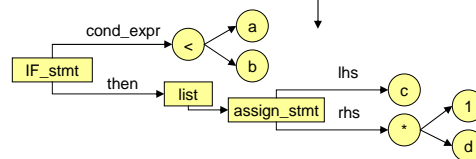
- ❑ High level program \rightarrow token strings
- ❑ E.g., IF (a<b) THEN c=1*d;

Major contribution in compiler is to formalize these two steps
 \Rightarrow highly systematic analysis
 \Rightarrow much easier logic
 \Rightarrow potentially more efficient



❖ Syntax analyzer (parser)

- ❑ Match and recognize token sequences according to the grammars
- ❑ Organize the tokens into the parse tree (abstract syntax tree)



Phases of Modern Compilers

❖ Semantic analysis

- Traverse the parse tree
- Check to ensure that elements fit together meaningfully
 - Most common is type checking
 - In other applications, this may involve more advanced checking
- Generate IR

❖ Code optimization

- Optimize code such that it reduces resource consumption
 - E.g., Runs faster
 - E.g., Uses less memory
- This phase refers to IR code optimization
 - Machine independent
 - E.g., $X = Y * 0 \Rightarrow X = 0$

Phases of Modern Compilers

❖ Code generation

- Translate IR to machine code (instruction selection)
- Memory assignment
- Resource assignment (registers, I/O)
- Machine code optimization
 - E.g., Use registers for repeatedly referenced variables

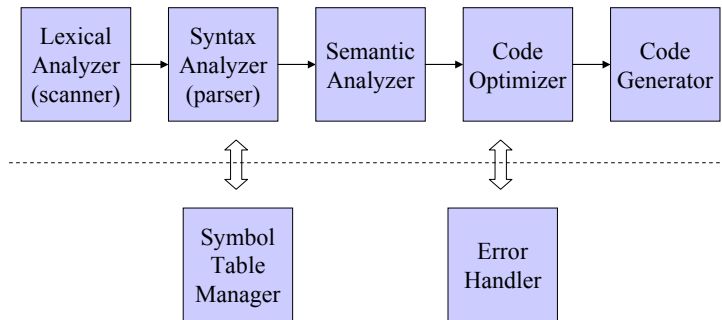
Symbol Table Management

- ❖ Symbol table is involved in all phases of the compiler process
 - Created at the lexical analysis phase
 - Information added at the syntax and semantics analysis phases
 - Used by all phases
 - Also used by debugger

Error Handler

- ❖ Error handler is also involved in all phases of the compiler process
 - Not of theoretical importance
 - But all practical systems should have error handling and reporting feature
 - Not covered in the course

Compiler Components -- Summary



Project Outline

