CSE 3302 Notes 1: Introduction

(Last updated 9/5/12 2:52 PM)

Got the CD?

Did anyone crash Linux in the browser?

Are you better off for having taken CSE 2312?

1.1. The Art of Language Design

Early IBM Tradition

Embedded Systems Attitude

> We'll never code in C, we need assembler.
>
> We'll never code in Java, we need C++.

The inevitability of converting code . . .

Why so many languages? (three answers in book)

What makes a language popular? (seven answers in book)

Specific concepts and features are more useful that bundles.

How about domain-specific languages and end-user programming?

1.2. The Programming Language Design Spectrum

Turing Award Winners with Strong Connections to PLs:

Year	Name	Contribution
1971	McCarthy	LISP
1972	Dijkstra	Philosophy
1977	Backus	FORTRAN, formalisms
1979	Iverson	APL
1980	Hoare	CSP
1983	Thompson	UNIX, scripting
1983	Ritchie	UNIX, C
1984	Wirth	Philosophy
1987	Cocke	RISC, optimization
1991	Milner	ML, inference

2001	Dahl	Simula, O-O
2001	Nygard	Simula, O-O
2003	Kay	O-O, SmallTalk, MVC
2005	Naur	Algol 60
2006	Allen	FORTRAN optimization
2008	Liskov	Abstraction, distributed computing

GCD Example:

C, Scheme, SML, Prolog, JavaScript code on webpage

Questions:

2320: Where do want to be in 10 years?

What is design? (juggling correctness, resource requirements, development cost)

3302: To get ahead, what do you want to *manipulate*?

First Class: Passed as argument, returned from function, assigned to variable, (Constructible?)

Second Class: Passed as argument

Third Class: None of the above

How about functions, threads/processes, processors, messages, channels/pipes?

Aside: COBOL - Can ALTER the destination of a GOTO . . .

1.3. Why Study Programming Languages?

ABET CAC Program Criteria for Computer Science:

Coverage of the fundamentals of . . . programming languages

ABET EAC Program Criteria for Software Engineering:

The curriculum must provide both breadth and depth . . . computer science topics

To be better now and in the future (book, pages 15 and 16)

Programming language application and PL implementation skills are a continuing distinguishing characteristic of computer science.

1.4. Compilation and Interpretation

Practice vs. Possibilities vs. Details

Assembler:

> Op Codes
> Labels
> Data Types (corresponding to hardware capabilities)
> Macros
> Control of Assembly

Compiler:

> Maximizes checking that can be performed without execution of the source program.
> May produce code for a machine whose level of similarity to ideal "language machine" may vary.

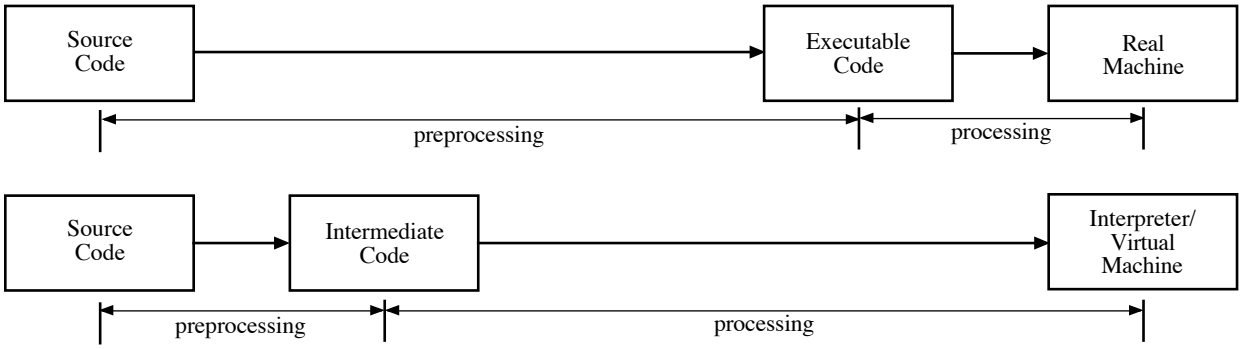
Interpreter:

> Ranges from general hardware machine to "language machine"

Linker/Loader:

> Resolves external references in several object files
> Possibly commence execution or just produce executable

From D. Grune, et.al., *Modern Compiler Design*, Wiley, 2000.



1.6. An Overview of Compilation

Preprocessor (e.g. C)

- Include files
- Macros
- Conditional text
- Compiler directives

Scanning/Lexing/Tokenizing/Lexical Analysis

- Remove comments / white space
- Collect "minimal" meaningful substrings
  - Identifiers/reserved words
  - Operators
  - Constants
- Strong connections to regular expressions and finite-state automata (CSE 3315)

Parsing/Syntax Analysis

- Construct tree representing nesting structure of language constructs and expressions
- Practical languages allow both bottom-up and top-down (e.g. recursive descent) approaches

Symbol Tables

- Data structures
- Scope/namespaces

Semantic Analysis

- Attribute grammar evaluation - functions for sending semantic information through abstract syntax tree
- Type inference

Intermediate Code Generation

- Triples or quads - high-level machine language

Code Improvement

- Peephole optimization
- Basic Blocks:
  - Common subexpressions
  - Constant propagation
- Inlining
- Flow analysis - matching variable definitions (assignments) with uses
- Loop dependence analysis - can loop iterations run independently (or be modified to allow)?
- Subscript aliasing - can two subscripting expressions refer to the same location?
- Concurrentization
- Register usage