Compiler Design

Amey Karkare and Subhajit Roy
Department of Computer Science and Engineering
IIT Kanpur
Karkare @iitk.ac.in subhajit@iitk.ac.in

1

Acknowledgements

- Most of the text in the slide is based on classic text **Compilers: Principles, Techniques, and Tools** by **Aho, Sethi, Ullman** and **Lam**
- Slides are modified version of those created by Prof S K Aggarwal, IITK

2

Motivation

- Language processing is an important component of programming
- Many systems software and application programs require structured input
  - Operating Systems (command line processing)
  - Databases (Query language processing)
  - Type setting systems like Latex
  - Software quality assurance and software testing

3

Motivation

- Wherever input has a structure one can think of language processing
- Why study compilers?
  - Compilers use the whole spectrum of language processing technology

4

What do we expect to achieve by the end of the course?

- Knowledge to design, develop, understand, modify/enhance, and maintain compilers for (even complex!) programming languages
- Confidence to use language processing technology for software development
- Become a better programmer and better software developer

5

Organization of the course

- Assignments 15%
- Mid semester exam 20%
- End semester exam 30%
- Course Project 35%
  - Group of max 4 students
    - If you cannot find 3 partners, work with fewer

6

Important

- Last date to drop the course
  - Jan 30, 2022
- Different from the Academic Calendar
- No instructor approval for drop after the above date

7

Fix Time Zone and Name on CANVAS

- See the Announcement Section
- The default time zone in Canvas is USA time zone. Change it to Indian time zone
- Change your "names" to the format: **RollNo FirstName LastName**
- Full Name, Display Name, Sortable Name - ALL THREE should have the same format.

8

Expectations? Queries?

9

Bit of History

- How are programming languages implemented? Two major strategies:
  - Interpreters (Less studied)
  - Compilers (very well understood with mathematical foundations)
- Some environments provide both interpreter and compiler. Lisp, scheme etc. provide
  - Interpreter for development
  - Compiler for deployment
- Java
  - Java compiler: Java to interpretable bytecode
  - Java JIT: bytecode to executable image

10

Some early machines and implementations

- IBM developed 704 in 1954. All programming was done in assembly language. Cost of software development far exceeded cost of hardware. Low productivity.
- Speedcoding interpreter: programs ran about 10 times slower than handwritten assembly code
- John Backus (in 1954): Proposed a program that translated high level expressions into native machine code. Skepticism all around. Most people thought it was impossible
- Fortran I project (1954-1957): The first compiler was released

11

Fortran I

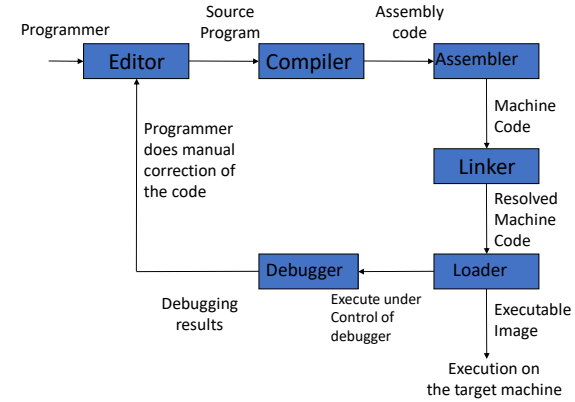
- The first compiler had a huge impact on the programming languages and computer science. The whole new field of compiler design was started
- More than half the programmers were using Fortran by 1958
- The development time was cut down to half
- Led to enormous amount of theoretical work (lexical analysis, parsing, optimization, structured programming, code generation, error recovery etc.)
- Modern compilers preserve the basic structure of the Fortran I compiler !!!

12

The big picture

- Compiler is part of program development environment
- The other typical components of this environment are editor, assembler, linker, loader, debugger, profiler etc.
- The compiler (and all other tools) must support each other for easy program development

13



21