Lecture 01 (Introduction)

1. Two or three days a week proposed
2. **CP is there**

1 Marks Distribution

1. Assignments - 20
2. CP - 20
3. Minor - 20
4. Major - 40

2 History

1. FORTRAN optimising compiler was polled to be one of the most influential inventions in a poll conducted in 2000
2. It was a trivial translator from FORTRAN to assembly + simple optimisations
3. Met with huge opposition :)
4. Hand written assembly is still faster than compiled code (opposition had some validity)
5. Compilers were not open-source initially and were sold around

3 GCC and LLVM

1. GCC is GPL license - copy-left license
    i. Copy-left is left recursive license
    ii. License is imported from the parent
    iii. Type of license cannot be modified
2. LLVM was graduate student project in UIUC - Apache license

4 Issues with Compilers (and Design)

1. GCC code base is increasing at the rate of 3000-4000 lines per year (double derivative ≥ 0)
2. LLVM also has a linear increase with a similar slope
3. Compiler design will be cumbersome and unattractive for new people

5 Points of Concern

1. Time
2. Power
3. Energy (technically dependent on previous two)

6 Setting the Stage

1. A program can be viewed as a function that takes an input and gives an output
2. Alternate *interpretation* is that a program can be *interpreted*

6.1 Interpreter vs Compiler

6.1.1 Interpreter

1. Interpreter is like a state machine which modifies the state of the data based on the instruction

$$(source, input) \rightarrow output \equiv source \rightarrow (input \rightarrow output)$$

2. Interpreter *interprets* line by line

6.1.2 Compiler

1. Compiler takes source program as input and gives the executable as output

$$source \rightarrow (input \rightarrow output) \equiv source \rightarrow executable$$

2. Compiler compiles the entire code at once before making the executable
3. Hence compiler has a lot of scope for optimisation vs an interpreter
4. Compiler can also be viewed as specialised partial evaluator (where interpreter == evaluator)