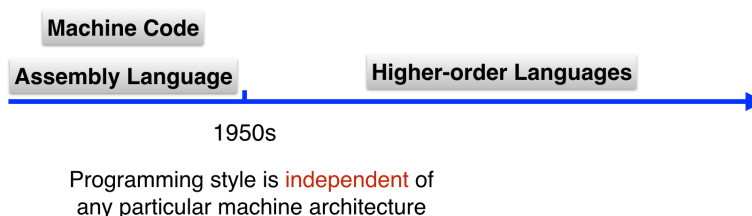


Overview of Programming Languages (I)

History of Programming Languages

- Earliest digital computers **without memory** (The imitation Game Clip “Turing’s Machine” 2014 <https://www.youtube.com/watch?v=nmXzPgVjxRw>)
 - adjust gears, connect cables and flip switches
- Later computers **with punch card reader** (Punch Card from a Fortran program https://en.wikipedia.org/wiki/Computer_programming_in_the_punched_card_era)
 - punch card: card-stock piece of paper with holes punched in it
 - data and instructions are holes
 - card reader reads holes
 - programmers **had to know the machine code** (different machines have different instructions)
 - *not easy to read and write*
- **Assembly language**
 - **machine-dependent**, CISC, RISC, close related to the machine architecture
 - the first abstraction of machine code, using text mnemonics to represent binary instructions and symbols to represent binary sequence
 - *punch card reader became interpreters*
 - more readable than pure machine code
 - **inefficient and error-prone** (no efficient flow control but JMP to lines)



- After 1950s
 - PL were intended to **bridge the gap** between natural language and the machine instructions
 - **Higher-ordered languages (our focus in CS333)**
 - Independent of any particular machine architecture
 - Closer to natural language
 - *Compilers/interpreters* translate the programs into assembly languages/machine code
-

Early PL

- developed to **satisfy particular needs**
- **influenced later language** development
- some **survived** and heavy used, some evolved or **replaced** by other languages (why)
 - too complex implement, too slow, or not general enough caused the language not survived
- some early PLs:
 - Fortran: scientific computing
 - long term used
 - **fast**
 - Cobol: business computing
 - long term used
 - **user friendly**
 - Algol: general purpose programming
 - short term
 - Lisp: AI programming
 - long term
 - **functional PL, 67 years old, treat computation as the evaluation of mathematical function, avoid changing status and mutable data**
 - C: system programming
 - long term
 - **explicit memory management**
 - Prolog: theorem proof
 - long term
 - SEQUEL: database management
 - SQL