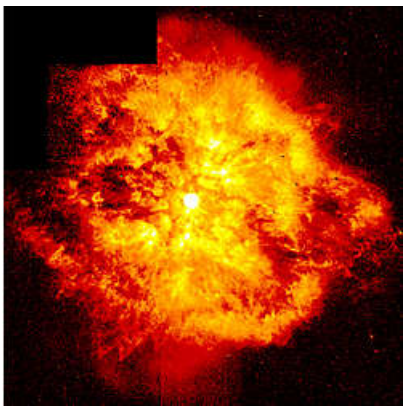
Evolution of Programming Languages

- Computers became available in the late 1940s and early 1950s, thanks to architecture designed by Hungarian genius John Von Neumann[1].
- They were slow, unreliable, expensive and **lacked any supporting software**. Hence the problem: **how shall we program them?**

1GL (1st Generation Language) – Machine Code

- Program computers directly in their "native" language!
- The "words" of a machine language are called *instructions*; each of these gives a basic command to the CPU. A program is just a long list of instructions that are *executed* by a CPU.
- Instructions are simply a pattern of bits. Different patterns correspond to different commands to the machine.
- Example command: 100010 01110 01001 01010 10000 001011
- Obvious problems: tedious, error-prone, hard to read, slow, difficult to modify, difficult to debug, etc.

We obviously need something better!



2GL (2nd Generation Language) – Assembler

- Although ***machine code*** may seem similar to ***assembly language*** they are in fact two different types of languages. ***Assembly code*** consists of both binary numbers and simple words whereas *machine code* is composed only of the two binary digits 0 and 1.
- It can be thought of as a human-readable notation for the machine language.
- Machine language, a mere pattern of bits, is made readable by replacing the raw values with symbols called ***mnemonics***.
- Example: "10110000 01100001" would now be written as "`mov al, 0x61`"

- There is usually a distinct correspondence between simple assembly and machine language.
- Transforming assembly into machine languages is accomplished by an assembler, the other direction by a disassembler.

- Basic Assembly Operations:
  - **a.** Moving
    1. load a value into a register
    2. move data from a memory location to a register, or vice versa
    3. read and write data from hardware devices
  - **b.** Computing
    1. add, subtract, multiply, or divide the values of two registers, placing the result in a register
    2. combine two register values with logical and/or
    3. negate a register value arithmetically or by logical *NOT*

[1] See http://ei.cs.vt.edu/~history/VonNeumann.html for more info on Von Neumann. Also see http://en.wikipedia.org/wiki/Von_Neumann_architecture for details on the architecture.

c. Affecting Program Flow
   1. jump to another location in the program (normally, instructions are processed sequentially)
   2. jump to another location, but save the next instruction as a point to return to
   3. go back to the last return point

- Problems: still too low-level for most programming tasks, messy, not portable enough

3GL (Third Generation Languages) – High Level Programming Languages

- more user-friendly, to some extent platform-independent, and abstract from low-level computer processor operations such as memory accesses.
- The word "high" does not mean better but rather refers to a high level of abstraction from machine language.
- See http://www.oreilly.com/pub/a/oreilly/news/languageposter_0504.html for the language poster.
- **Hall of Famers**:
  - the first high-level language, FORTRAN [*FOR*mula *TRAN*slation], was developed for scientific and engineering applications about 1956 by John Backus at the IBM Corp.
  - A program that handled recursive algorithms better, LISP [*LIS*t *P*rocessing], was developed by John McCarthy at the Massachusetts Institute of Technology in the early 1950s; implemented in 1959, it has become the standard language for the artificial intelligence community.
  - COBOL [*CO*mmon *B*usiness *O*riented *L*anguage], the first language intended for commercial applications, is still widely used; it was developed by a committee of computer manufacturers and users under the leadership of Grace Hopper, a U.S. Navy programmer, in 1959.
  - ALGOL [*ALGO*rithmic *L*anguage], developed in Europe about 1958, was used primarily in mathematics and science
  - ADA [for Ada Augusta, Countess of Lovelace, biographer of Charles Babbage and the first programmer!], developed in 1981 by the U.S. Dept. of Defense, was designed for both business and scientific use.
  - BASIC [*B*eginner's *A*ll-purpose *S*ymbolic *I*nstruction *C*ode] was developed by two Dartmouth College professors, John Kemeny and Thomas Kurtz, as a teaching tool for undergraduates (1966); it subsequently became the primary language of the personal computer revolution.
  - In 1972, to implement the UNIX operating system, Dennis Ritchie and Brian Kernighan of Bell Laboratories produced a language that he called C. Along with its popular object-oriented extension C++, developed by Bjarne Stroustrup of Bell Laboratories in 1979, it has become the most widely used general-purpose language among professional programmers.
  - Java is an object-oriented language similar to C++ but simplified to eliminate features that are prone to programming errors. Java was developed specifically as a network-oriented language, for writing programs that can be safely downloaded through the Internet and immediately run without fear of viruses. Using small Java programs called applets, WWW pages can be developed that include a full range of multimedia functions.

4GLs

nonprocedural—they specify what is to be accomplished without describing how.

5GLs

Focus is on Artificial Intelligence (AI) (e.g. Prolog, LISP)

"Special" Languages:

- GPSS [*G*eneral *P*urpose *S*ystem *S*imulator] is used for modeling physical and environmental events.
- SNOBOL [*S*tring-*O*riented *S*ymbolic *L*anguage] is designed for pattern matching and list processing.
- PILOT [*P*rogrammed *I*nstruction *L*earning, *O*r *T*esting] is used in writing instructional software.
- Occam is a non-sequential language that optimizes the execution of a program's instructions in parallel-processing systems.