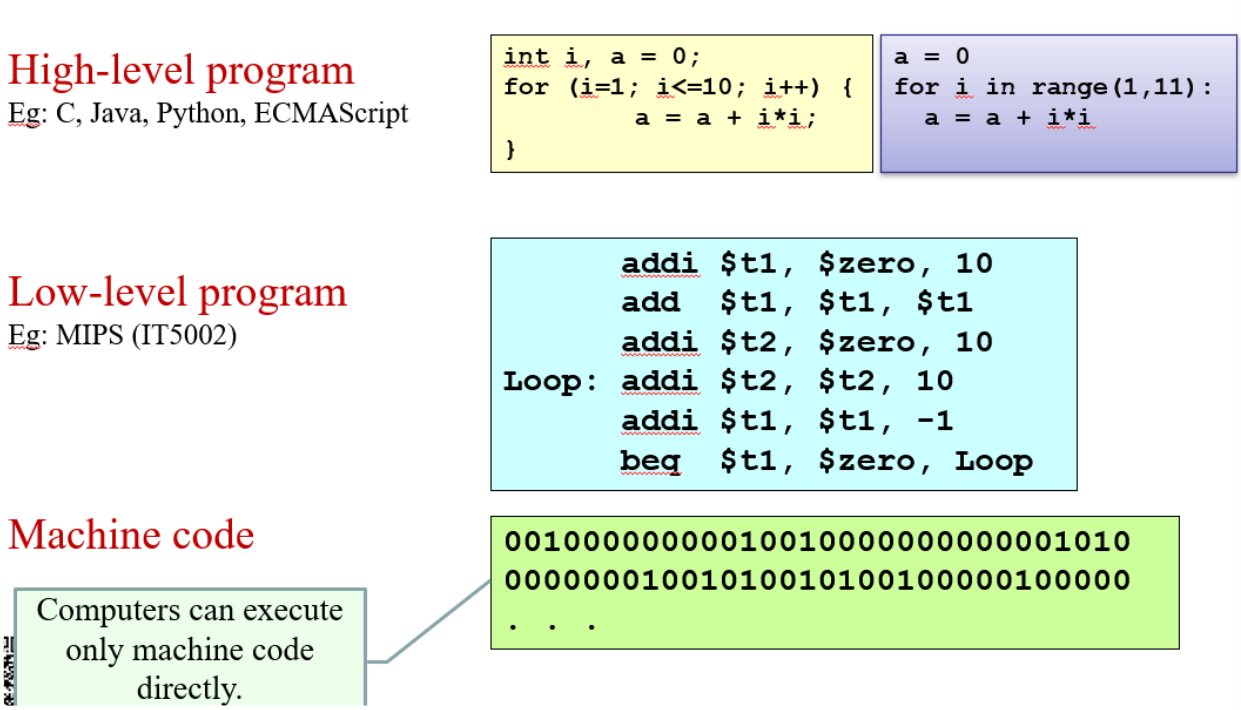
1 - Introduction

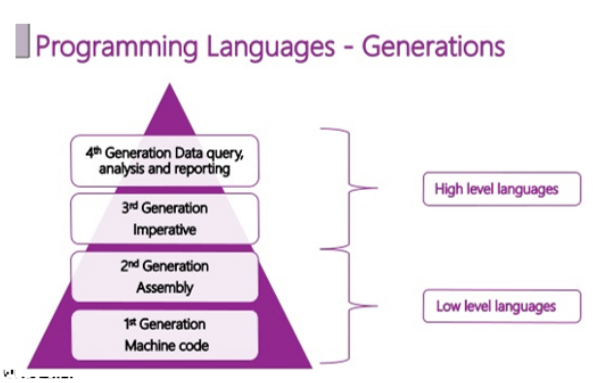
1.1 Programming Language

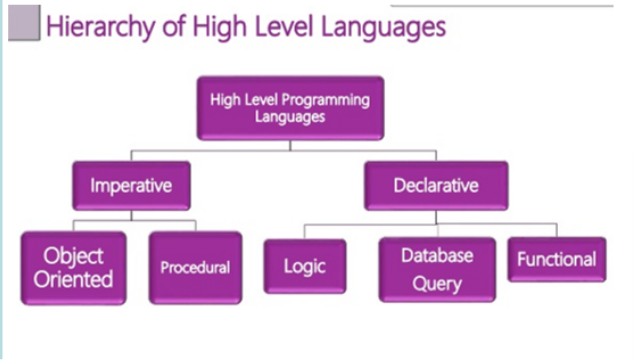
- Programming language: a formal language that specifies a set of instructions for a computer to implement specific algorithms to solve problems



- 1st Generation languages:
  - Machine language
  - Directly executable by machine
  - Machine dependent
  - Efficient code but difficult to write
- 2nd generation languages:
  - Assembly language
  - Need to be **translated(assembled)** into machine code for execution
  - Efficient code, easier to write than machine code
- 3rd generation language:
  - Closer to English
  - Need to be **translated (complied or interpreted)** into machine code for execution
  - Example: FORTRAN, COBOL, C, BASIC
- 4th generation language:
  - Require fewer instructions than 3GL
  - Used with databases (query languages, report generators, forms designers)
  - Example: SQL, PostScript, Mathematica
- 5th generation language:
  - Used mainly AI research

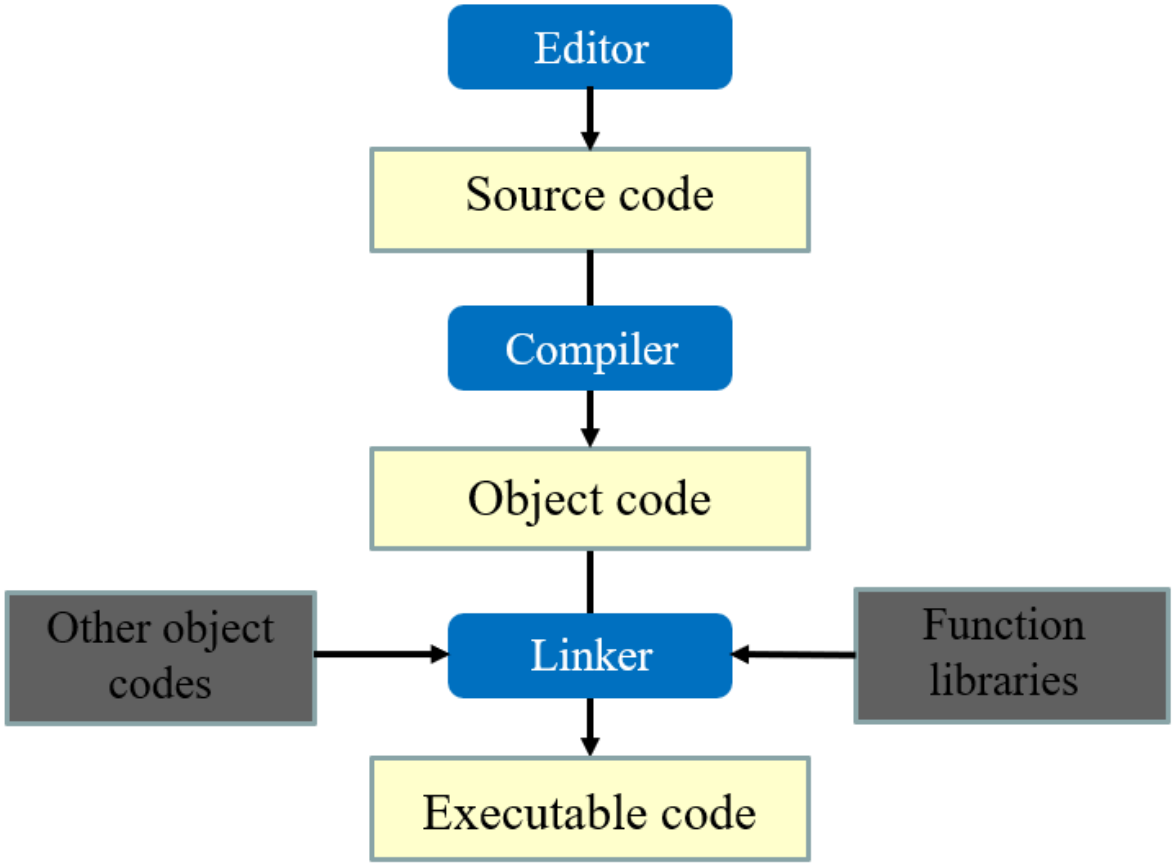
    - Used mainly AI research
    - Declarative languages
    - Functional languages (Lisp, Scheme, SML)
    - Logic programming (Prolog)
- "Generational" classification of high level languages (3GL and later) was never fully precise
- A different classification is based on paradigm





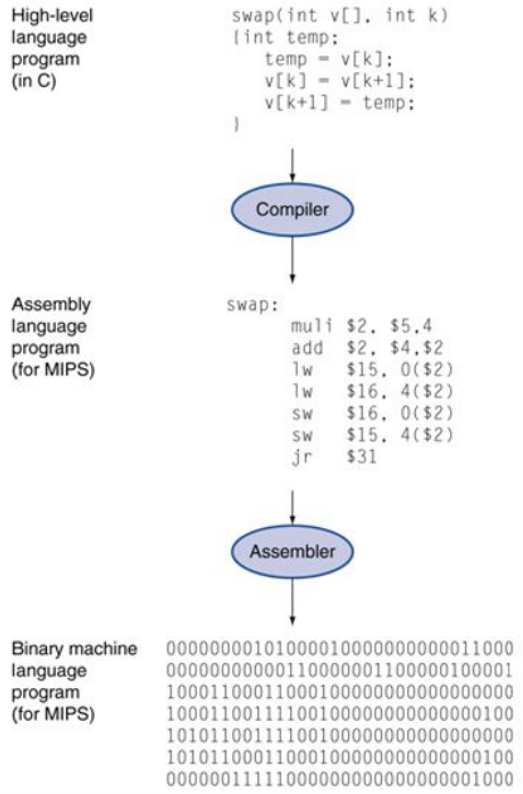
1.1.1 C Programming Language

- C is an **imperative procedural language (命令式程序语言)**
- C provides constructs that map efficiently to typical machine instructions
- C is a high-level language very close to the machine level, hence sometimes it is called "mid-level"

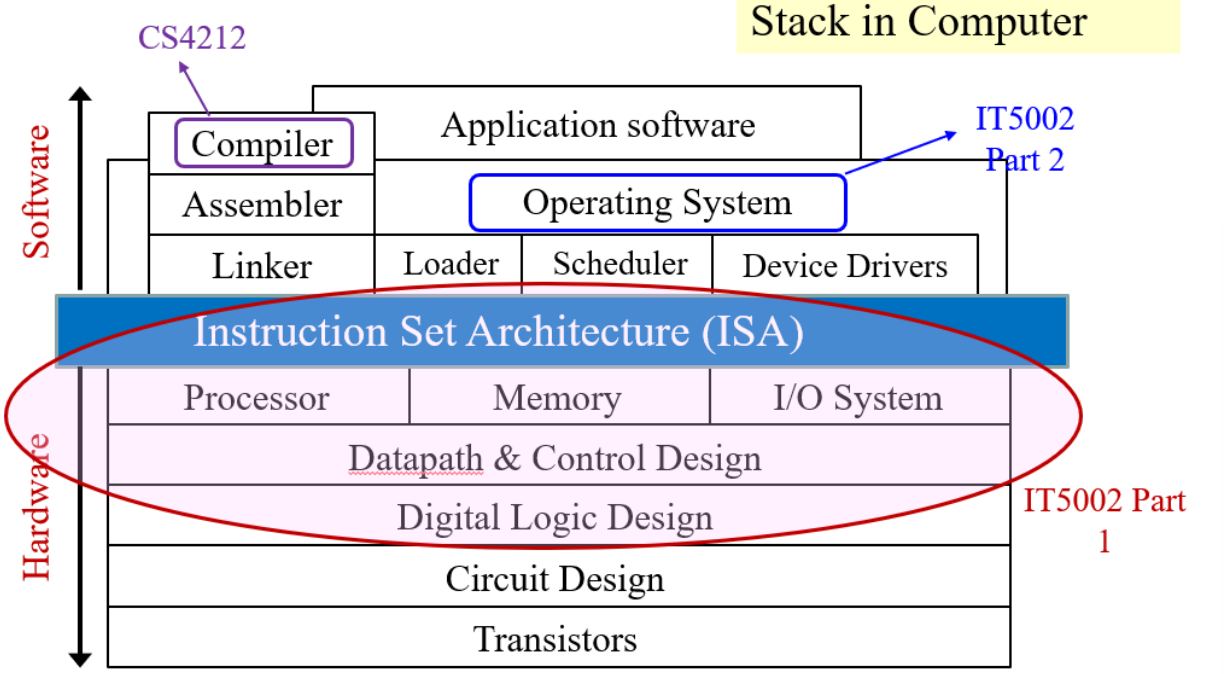


1.2 Abstraction

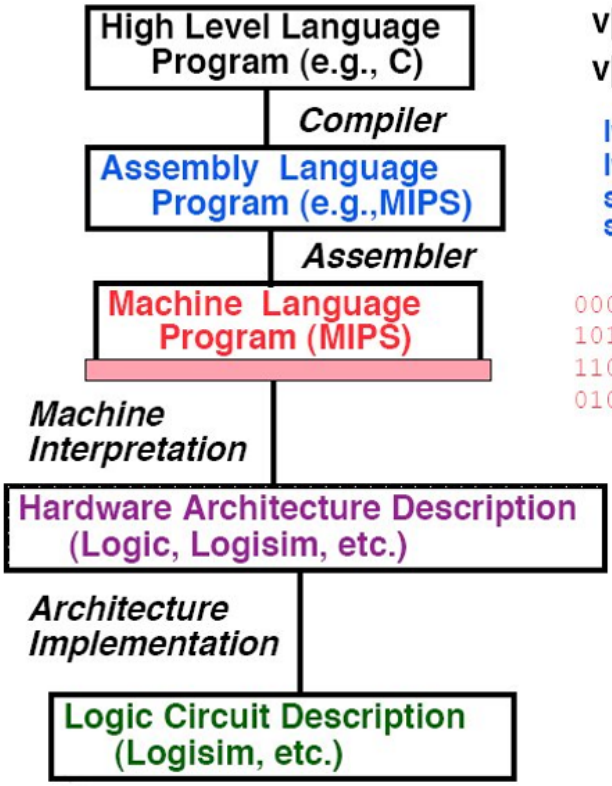
- High-level language
  - Level of abstraction closer to problem domain
  - Provides productivity and portability
- Assembly language
  - Textual and symbolic representation of instructions
- Machine code (object code or binary)
  - Binary bits of instructions and data



Hardware/Software Stack in Computer



Level of Representation




```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw   $t0, 0($2)
lw   $t1, 4($2)
sw   $t1, 0($2)
sw   $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```