Programming Languages

Low-level Languages

High-level Languages

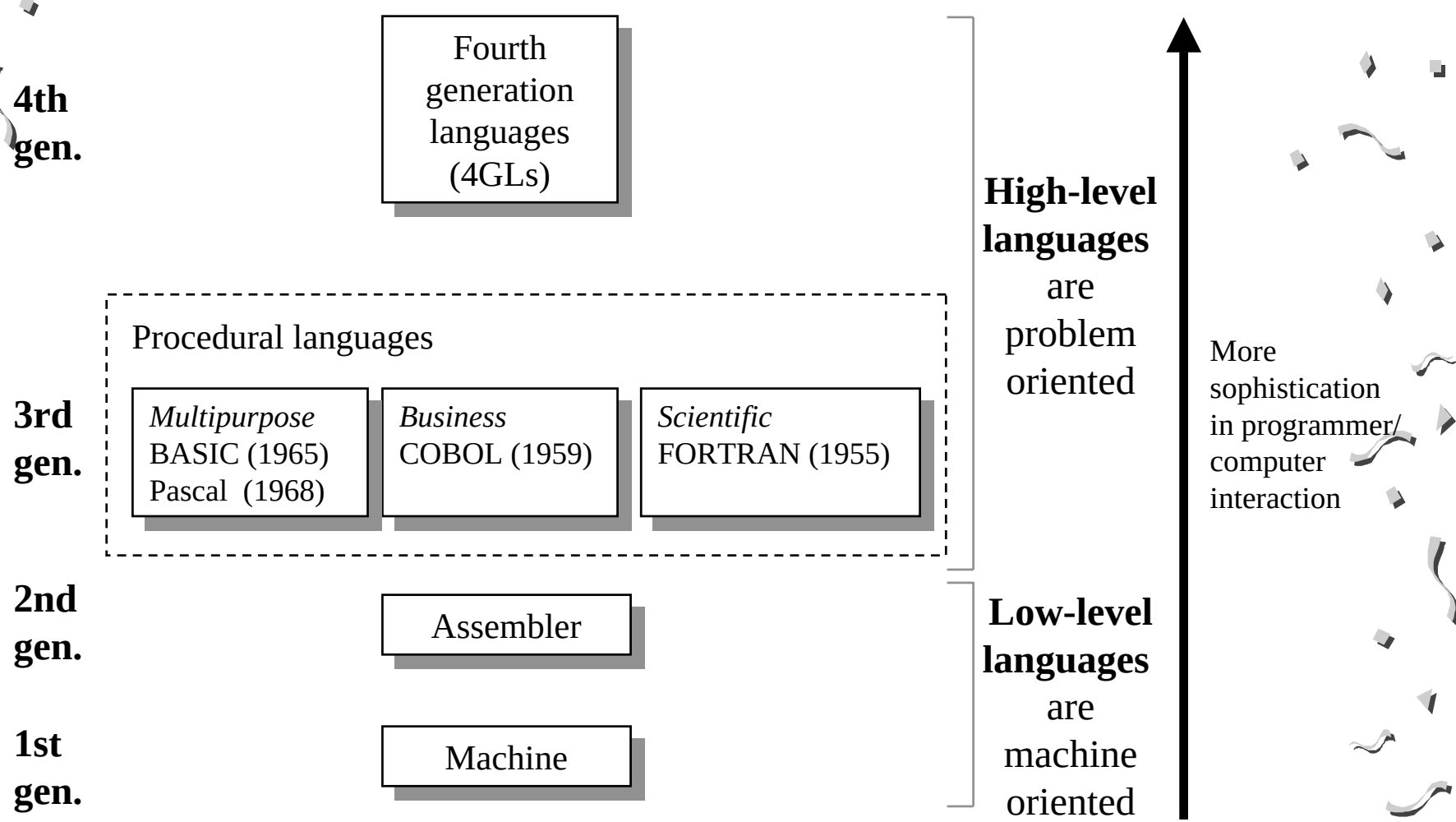
Advantages & Disadvantages

Translators

Trends in Programming Languages

Introduction

- program
  - a series of instructions make a computer ‘works’
  - can be written in a variety of programming languages
- programming lang. - later generations:
  - fewer instructions
  - provide a more sophisticated programmer/computer interaction



The hierarchy of programming languages

Low-level Languages

- Machine language
  - a programming language that is interpreted & executed directly by the computer
- Assembly language
  - a symbolic language with an instruction set that is basically one-to-one with the machine language

Machine language (ML)

- each computer can only understand programs that are written in its own ML
- is provided by the computer manufacturer
- translation is needed when executing programs written in Pascal or BASIC
- written at the most basic level of computer operation
  - coded as a series of 0's and 1's, e.g., 10111010

Machine language (ML)

- Disadvantages:
  - very difficult to write, because:
    - binary system - not ‘user friendly’ to human
    - it requires excellent memorising power
  - programmer has to keep track of storage locations of data & instruction

⇨ *ASSEMBLY LANGUAGE*

Assembly language

- also provided by the manufacturer
- one instruction for each computer operation
- instruction codes are represented by ***mnemonics*** (a set of letters)
- the code must be assembled into machine language for execution
  - ☆ assembly language

[refer to Fig 13.2]

Assembly language

- Disadvantages: (similar to that of ML)
  - machine dependent
  - the program is usually long
  - hard to learn & slow to write

⇨ *HIGH LEVEL LANGUAGES*

High-level Languages

- made programming much more convenient
- written using common names & words
  - more like human languages
- problem-oriented languages
  - designed to solve specific problems
- e.g., FORTRAN, COBOL, BASIC, Pascal & C language

[refer to Fig 13.4 - 13.8]

Advantages of High-level Languages

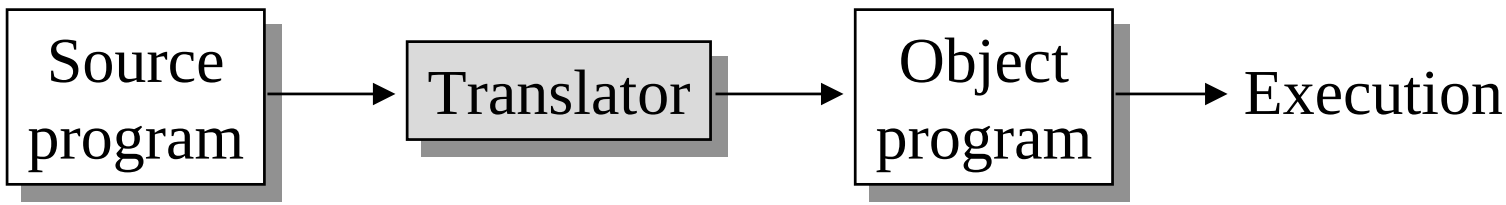
- easier to write, to read & to modify
  - written in English-like format
- programs - faster & shorter to code
  - one statement for several computer operations
- more portable, i.e., can be executed by different computers
  - machine independent

Advantages of Low-level Languages

- take up less storage space
- run faster
- useful for writing system programs
  - e.g., operating systems (require fast & efficient use of CPU)
- sometimes an operation can only be performed in a low-level language

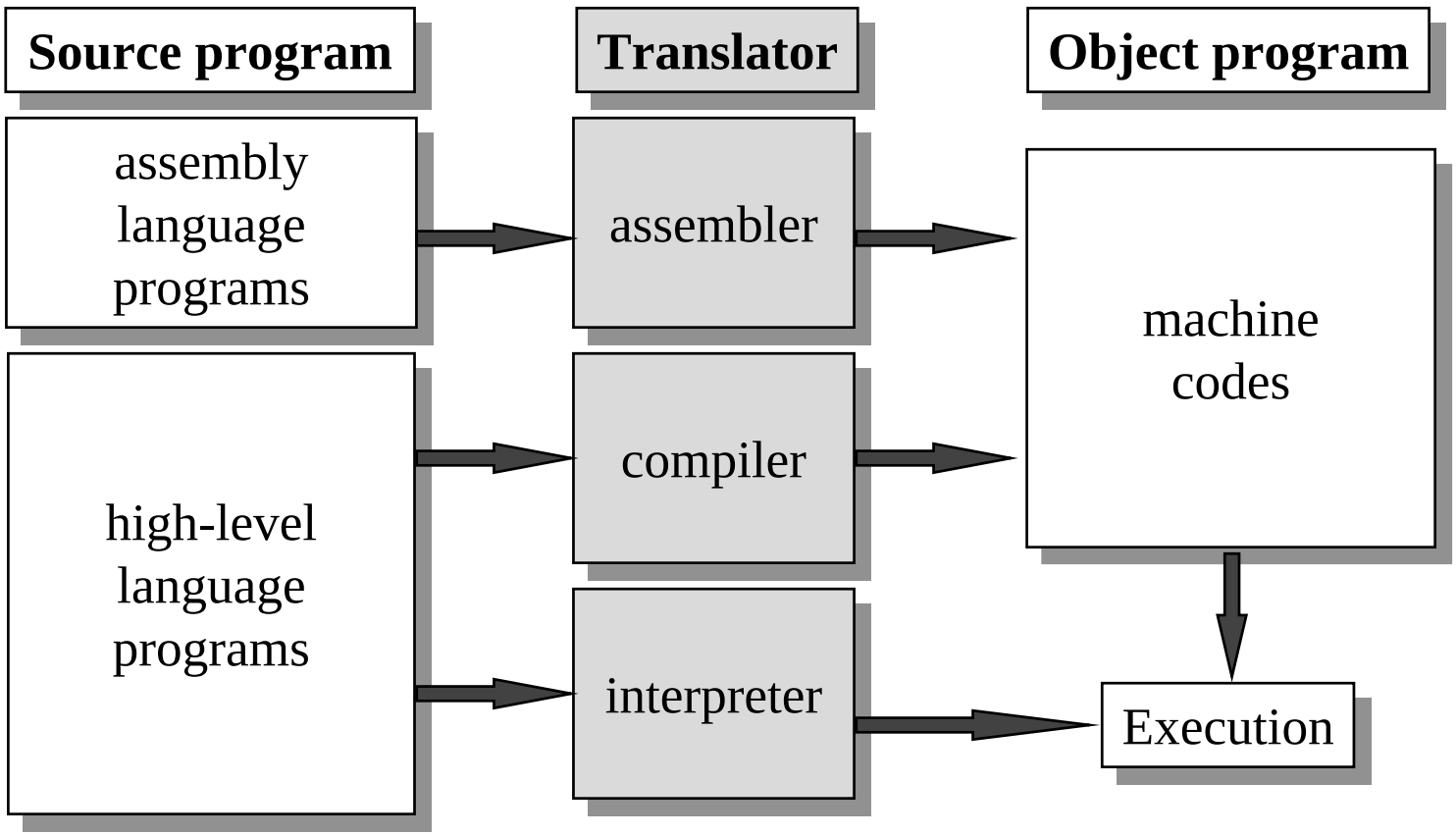
Translators

- programs must be translated into machine codes before execution



A simplified translation process

Translators - 3 types



The functions of the three types of translators

Similarities between compilers & interpreters

- both translate high-level languages to machine codes
- both direct errors in the programs & print error messages
- both work out where to store the object program & its data in the memory

Differences between compilers & interpreters

	Compilers	**Interpreters**
Translation of source program	the whole program before execution	one line at a time when it is run
Freqrency of translation	each line is translated once	has to be translated every time it is executed - slower
Object program	can be saved for future exeution without the source program	no object program is generated, so, source program must be present for execution

Trends in Programming Languages

- fewer instructions
- more user-friendly
- towards using 4GLs
  - non-procedural language
  - users only have to state what needs doing, but not how to do it
  - designed for users with minimal programming knowledge & training

Trends in Programming Languages

- Logic programming (e.g., Prolog)
  - 5th-generation computer language
  - declarative language
  - expert systems & artificial intelligence app.
- Object-oriented techniques
  - e.g., Borland C++
- Visual development environments
  - e.g., Microsoft Visual Basic