

01 Introduction

Instructor: Ke Wei (柯韋)

➡ A319 ☎ Ext. 6452 ✉ wke@ipm.edu.mo

<http://brouwer.ipm.edu.mo/COMP112/18/>

Bachelor of Science in Computing, School of Public Administration, Macao Polytechnic Institute

August 27, 2018



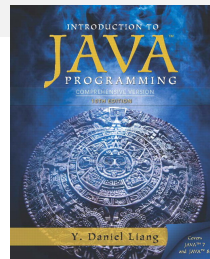
Textbooks and References



Y. Daniel Liang.

Introduction to Java Programming, Comprehensive Version, 10th Edition.
Pearson, 2014.

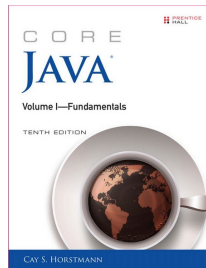
ISBN 9780133761313 *Textbook*.



Cay S. Horstmann.

Core Java Volume I — Fundamentals, 10th Edition.
Prentice Hall, 2016.

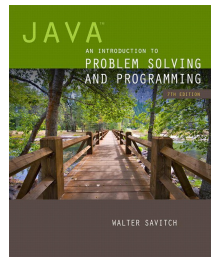
ISBN 9780134177304 *Reference book*.



Walter Savitch.

Java: An Introduction to Problem Solving and Programming, 7th Edition.
Pearson, 2014.

ISBN 9780133766264 *Reference book*.



Outline

- 1 Textbooks and References
- 2 Course Overview
- 3 Computer Architecture
- 4 Instructions and Programs
- 5 Reading Homework

Course Overview

This is an introduction to computer programming and the Java language fundamentals, including:

- the concept and history of programming,
- compiling and running programs,
- calling functions and methods,
- using variables and expressions,
- writing conditional statements,
- writing loops and methods, and
- using arrays and objects to group data items.

A Simple Program

- Here is a simple program expressed in natural language.

$$\left\{ \begin{array}{l} \bullet \text{ input a number and store it in variable } x \\ \bullet \text{ if } x \text{ is greater than or equal to } 0 \text{ then} \\ \quad \left\{ \begin{array}{l} \bullet \text{ compute the result of } \sqrt{x} \text{ and store it in variable } y \\ \bullet \text{ print the number stored in } y \end{array} \right\} \\ \text{otherwise} \\ \quad \left\{ \bullet \text{ print "Error: the input number is negative."} \right\} \end{array} \right\}$$

- The main part of a program is a *block of statements*, listed sequentially.
- There can be blocks nested inside blocks. Some blocks of statements are conditional.
- A program accepts input and produces output.
- Values are stored in *variables*, and **computations** can use the values in the variables.

Course Sections

The course is divided into the following sections:

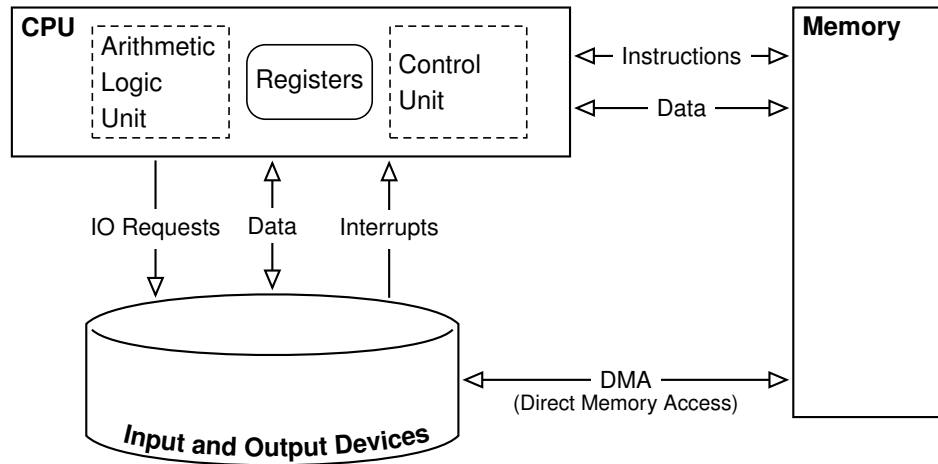
- architecture of computers and introduction to programming languages,
- compiling and running Java programs (with and without an IDE),
- language syntax and program structure,
- variables and data types,
- expressions and statements,
- simple graphics,
- control flow, conditionals and loops,
- grouping variables in arrays and using arrays with loops,
- grouping variables in classes and using instances of classes, and
- writing functions and procedures (static methods), passing parameters and returning results.

Functional Units

From the programming point of view, a computer can be divided into some functional units.

- ① Central processing unit (CPU)
- ② Memory, random-access memory (RAM) and read-only memory (ROM)
- ③ I/O devices

A Diagram of Computer Architecture



Functional Units — CPU

Arithmetic and logic unit (ALU)

- Executes arithmetic and logical operations, such as *addition* (+) and *conjunction* (&).
- Contains a number of very high speed storage elements, called *registers*, for holding *operands* and results.

Control unit

- Coordinates and transfers data among the functional units.
- Fetches *instructions*, as numbers, from memory and decodes the instructions.
- Directs operations to perform in the functional units according to the instructions.

A central processing unit (CPU) usually consists of the above two components.

- A CPU executes an instruction in one or a few clock ticks.
- The faster the clock speed, measured in frequency (hertz, Hz), the faster the CPU.

Functional Units — Memory

Memory

- Memory stores programs and data for later reference.
- Memory consists of semiconductor units, each unit stores one *bit*, either 0 or 1.
- Bits are processed together in groups of a fixed size, called *words*. The number of bits in a word is called the *word length*, which is CPU-dependent, usually 32 or 64.
- Each word stores a whole number in binary number system, ranging from 0 to $2^{\text{word length}} - 1$.
- The size of memory is measured in *bytes*, a group of eight bits, independent to the word length of a particular CPU.
- Each byte of memory has a unique location in the memory.
- The location of a byte is also a whole number, indicating at which place the byte is located from the beginning of the memory. This number is called the memory *address*.

Functional Units — I/O Devices

Input Devices

- Translates information obtained from the outside world to digital signals and provides them to computers.
- Provides data in device registers which are mapped to memory locations.

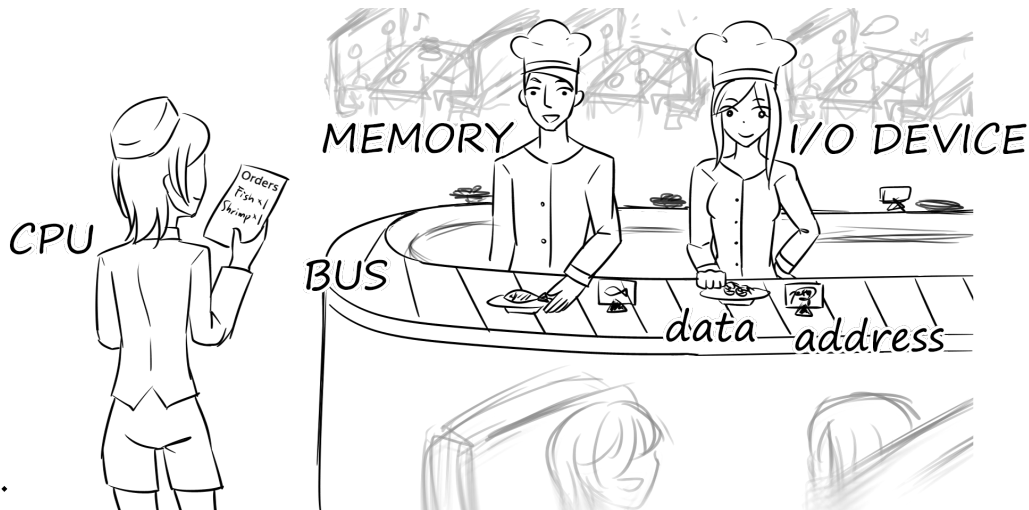
A read from the mapped memory location inputs the data from the input device.

Output Devices

- Translates digital signals generated by computers into desired formats: video, audio, text, radio, ..., and provides them to the outside world.
- Outputs data in device registers which are mapped to memory locations.

A write to the mapped memory location outputs the data to the output device.

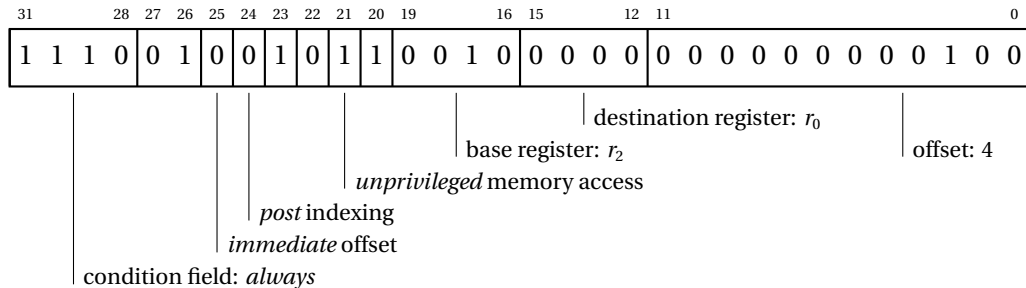
Communicating through Buses



Instructions and Programs

- Activities in a computer are governed by instructions.
- A group of well organized instructions forms a program.
- Typical computer instructions perform very simple operations in very high speed.
- Instructions are encoded as binary numbers. This form is called *machine language*.
- Programmers often write instructions in their symbolic form — using *mnemonics*. This form is called *assembly language*.
- An assembler translates a program written in assembly language to the form of machine language — *binary code*.
- Machine languages, assembly languages and assemblers are platform-dependent.
- Instructions are stored in memory along with data. This is known as the *Von Neumann architecture*.
- There is a special register called the *program counter (pc)* to mark the address of the current instruction to execute. The *pc* increments automatically.

The Binary Format of an Instruction



An ARM 32-bit instruction: `ldr r0,[r2],#4`

Fundamental Simple Instructions

These simple instructions include:

- transfer data between memory and registers,

$$r_0 \leftarrow [100] \quad [104] \leftarrow r_1 \quad r_0 \leftarrow r_2 \quad r_2 \leftarrow [r_0] \quad r_1 \leftarrow 100,$$

where a memory cell is denoted by its *[address]*,

- Sometimes, a small operand can be stored in the instruction directly, called an *immediate operand*.
- perform arithmetic and logical operations on the *operands* stored in registers and put the results back to registers,

$$r_0 \leftarrow r_1 + r_2 \quad r_0 \leftarrow r_0 - r_1 \quad r_2 \leftarrow r_2 \times 2,$$

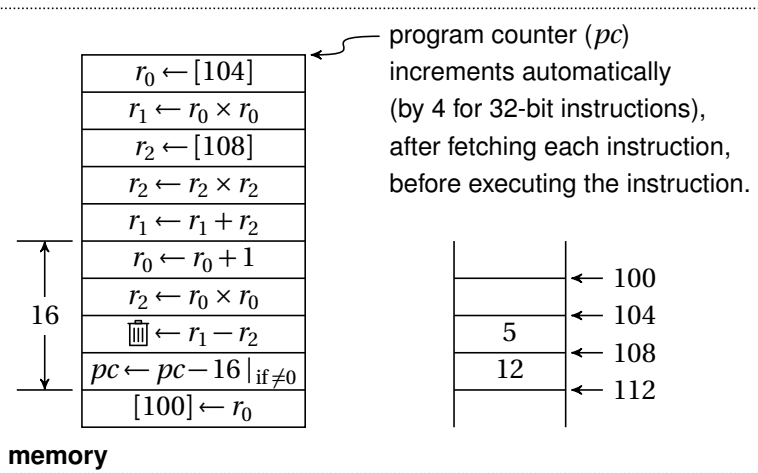
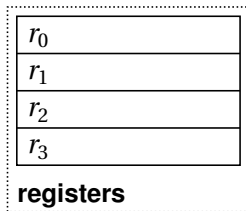
- control the execution flow of the instructions by changing the *pc*, often conditionally,

$$pc \leftarrow pc - 16 \mid_{\text{if} \geq 0}.$$

A Program Example


This program is to store an integer to memory [100] such that

$$[100]^2 = [104]^2 + [108]^2.$$



Reverse Engineering

$r_0 \leftarrow 0$
$r_1 \leftarrow [204]$
$r_1 \leftarrow r_1 \times 4$
$r_2 \leftarrow 208$
$r_1 \leftarrow r_1 + r_2$
$\text{[trash]} \leftarrow r_2 - r_1$
$pc \leftarrow pc + 24 \mid \text{if} \geq 0$
$r_3 \leftarrow [r_2]$
$r_2 \leftarrow r_2 + 4$
$\text{[trash]} \leftarrow r_3 - 0$
$pc \leftarrow pc - 24 \mid \text{if} \geq 0$
$r_0 \leftarrow r_0 + 1$
$pc \leftarrow pc - 32$
$[200] \leftarrow r_0$

 Try to figure out what the following program intends to compute.

r_0	r_1	r_2	r_3
0	40	208	—
0	248	208	—

Trace the contents of the registers using this table.

	← 200
	← 204
10	← 208
1	← 212
15	← 216
-4	← 220
-7	← 224
3	← 228
12	← 232
-8	← 236
0	← 240
6	← 244
-12	← 248
-10	

Reading Homework

Textbook

- Section 1.1 – 1.4.

Internet

- Instruction set ([http://en.wikipedia.org/wiki/Instruction_\(computer_science\)](http://en.wikipedia.org/wiki/Instruction_(computer_science))).

Self-test

- 1.1 – 1.10 (<http://tiger.armstrong.edu/selftest/selftest9e?chapter=1>).

