Chapter 1

Introduction to Computation

Algorithms, Processors, and Programs
Algorithm

- A sequence of unambiguous steps for accomplishing some task or process.
- In theory an algorithm can have an infinite number of steps but in practice algorithms must have a finite number of steps.
- Example: calculating $\sqrt{2}$ requires an infinite number of steps (infinite non-repeating decimal expansion).
Finite Algorithm for $\sqrt{2}$

\begin{align*}
x_0 &\leftarrow 1 \\
x_1 &\leftarrow \frac{x_0 + 2}{x_0} \\
x_2 &\leftarrow \frac{x_1 + 2}{x_1} \\
x_3 &\leftarrow \frac{x_2 + 2}{x_2} \\
x_4 &\leftarrow \frac{x_3 + 2}{x_3} \\
x_5 &\leftarrow \frac{x_4 + 2}{x_4} \\
\text{RETURN } x_5
\end{align*}
Processor

- Any device, calculator, computer, virtual machine, or human being that can process or execute an algorithm.
- Each type of processor understands a language (e.g., machine language) and algorithms must be expressed in this language in order for the processor to be able to execute them.
Memory words contain data. Each memory word has an address which is used to locate this data. A common size for a word is 8 bits (byte). One or more bytes is required to store each data item (an integer for example) or machine language instruction.

<table>
<thead>
<tr>
<th>Addresses</th>
<th>Memory Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001100...000</td>
<td>11100111</td>
</tr>
<tr>
<td>0001100...001</td>
<td>00000110</td>
</tr>
<tr>
<td>0001100...010</td>
<td>01011101</td>
</tr>
<tr>
<td>0001100...011</td>
<td>10010010</td>
</tr>
<tr>
<td>0001100...100</td>
<td>01110111</td>
</tr>
<tr>
<td>0001100...101</td>
<td>10100000</td>
</tr>
</tbody>
</table>
Block diagram of a computer
Program

- A representation of an algorithm as a sequence of instructions that can be understood and executed by a processor to complete the task, problem or process described by the algorithm.

- For a CPU the program is a sequence of machine language instructions stored in memory in binary form.
Computer Languages

- Machine language (Lowest level)
- Assembly language
  - mnemonic form of machine language
- High level language
  - C, C++, Java
  - Closer to the human level of problem solving
An addition problem

- **Pseudo-code**
  
  \[
  \begin{align*}
  i &\leftarrow 6 \quad \text{(assign 6 to i)} \\
  j &\leftarrow 7 \quad \text{(assign 7 to j)} \\
  k &\leftarrow 511 \quad \text{(assign 511 to k)} \\
  \text{sum} &\leftarrow i + j + k \quad \text{(assign sum to sum)}
  \end{align*}
  \]
Machine Language (PC)

Following string of 112 bits

1010  0001  0000  0000  0000  0000  0000  0000
0011  0000  0110  0000  0000  0000  0000  0010
0000  0011  0000  0110  0000  0000  0000  0000
0100  1010  0011  0000  0000  0000  0000  0110
Assembly language (PC)

```
mov    ax,    i
add    ax,    j
add    ax,    k
mov    sum,   ax

i        dw    6
j        dw    7
k        dw    511
sum      dw    0
```
Java, C++

```java
int i = 6;
int j = 7;
int k = 511;
int sum = i + j + k;
```
The compilation process

The compiler translates the high level language source code program into a machine language object code program that can be understood by the computer's CPU.
The interpretation process

The interpreter translates one statement at a time and has it executed by the CPU before returning to translate another statement.
The Java Virtual Machine

(a) Java Source Code → Java Compiler → Bytecode (class file)

(b) Bytecode (class file) → Java Interpreter → Java Virtual Machine
   → Real Machine
### Java translation example

<table>
<thead>
<tr>
<th>Assembly Language</th>
<th>Bytecode (hexadecimal)</th>
<th>Bytecode (binary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bipush 6</td>
<td>10 06</td>
<td>0001 0000 0000 0110</td>
</tr>
<tr>
<td>istore_1</td>
<td>3C</td>
<td>0011 1100</td>
</tr>
<tr>
<td>bipush 7</td>
<td>10 07</td>
<td>0001 0000 0000 0111</td>
</tr>
<tr>
<td>istore_2</td>
<td>3D</td>
<td>0011 1101</td>
</tr>
<tr>
<td>sipush 511</td>
<td>11 01 00 FF</td>
<td>0001 0001 0000 0001 1111 1111</td>
</tr>
<tr>
<td>istore_3</td>
<td>3E</td>
<td>0011 1110</td>
</tr>
<tr>
<td>iload_1</td>
<td>1B</td>
<td>0001 1011</td>
</tr>
<tr>
<td>iload_2</td>
<td>1C</td>
<td>0001 1100</td>
</tr>
<tr>
<td>iadd</td>
<td>60</td>
<td>0110 0000</td>
</tr>
<tr>
<td>iload_3</td>
<td>1D</td>
<td>0001 1101</td>
</tr>
<tr>
<td>iadd</td>
<td>60</td>
<td>0110 0000</td>
</tr>
<tr>
<td>istore 4</td>
<td>36 04</td>
<td>0011 0110 0000 0100</td>
</tr>
</tbody>
</table>