Overview of Sequential Programming Concepts

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Lesson

• Understand the meaning of key concepts associated with sequential programming
  • e.g., each step in a program is executed in order one at a time
Learning Objectives in this Lesson

- Understand the meaning of key concepts associated with sequential programming
  - e.g., each step in a program is executed in order one at a time

Mastering these concepts is essential before trying to learn more advanced concurrent & parallel programming concepts.
An Overview of Sequential Programming
An Overview of Sequential Programming

• Sequential programming is a form of computing that executes the same sequence of instructions & always produces the same results
  • i.e., execution is *deterministic*

See en.wikipedia.org/wiki/Sequential_algorithm
An Overview of Sequential Programming

- Sequential programming is a form of computing that executes the same sequence of instructions & always produces the same results.
- i.e., execution is deterministic.

Given a certain input, the same output will always be produced in the same order.

An Overview of Sequential Programming

• The deterministic behavior of sequential programs assumes no deliberate use of randomness, of course

See en.wikipedia.org/wiki/Randomized_algorithm
An Overview of Sequential Programming

• The deterministic behavior of sequential programs assumes no deliberate use of randomness, of course

See upcoming lessons on the Java Fork-Join framework for coverage of how randomness is applied in concurrent & parallel programs

An Overview of Sequential Programming

• Sequential programs have two main characteristics

See www.doc.ic.ac.uk/~jnm/concurrency/online/concurrent/tsld007.htm
An Overview of Sequential Programming

- Sequential programs have two main characteristics:
  - The textual order of statements specifies their order of execution

```java
public E get(int index) {
    rangeCheck(index);
    return elementData (index);
}
```
An Overview of Sequential Programming

- Sequential programs have two main characteristics:
  - The textual order of statements specifies their order of execution

```java
public E get(int index) {
    rangeCheck(index);
    return elementData(index);
}
```

E.g., chaos & insanity will occur in Java’s ArrayList get() method if rangeCheck() is not called before elementData()!!!
An Overview of Sequential Programming

- Sequential programs have two main characteristics:
  - The textual order of statements specifies their order of execution.
  - Successive statements must execute without any temporal overlap visible to programs.

Consider the code sequence:

\[
\begin{align*}
a &= b + c \\
d &= e - a
\end{align*}
\]
An Overview of Sequential Programming

- Sequential programs have two main characteristics:
  - The textual order of statements specifies their order of execution.
  - Successive statements must execute without any temporal overlap *visible* to programs.

Consider the code sequence:

\[
\begin{align*}
    a &= b + c \\
    d &= e - a
\end{align*}
\]

The value of ‘a’ must be assigned before the value of ‘d’ is assigned.
An Overview of Sequential Programming

• Sequential programs have two main characteristics:
  • The textual order of statements specifies their order of execution
  • Successive statements must execute without any temporal overlap visible to programs
  • However, lower layers in the solution stack can reorder instructions transparently

Consider the code sequence:

\[
\begin{align*}
a &= b + c \\
d &= e - a
\end{align*}
\]

An Overview of Sequential Programming

- Sequential programs have two main characteristics:
  - The textual order of statements specifies their order of execution.
  - Successive statements must execute without any temporal overlap visible to programs.
- However, lower layers in the solution stack can reorder instructions transparently.

Consider the code sequence:

\[
\begin{align*}
a &= b + c \\
d &= e - a
\end{align*}
\]

Assuming \(a, b, c, d, \) & \(e\) are in memory & loads/stores take one clock cycle out-of-order, then instruction scheduling eliminates pipeline stalls.

![Diagram of pipeline stages and clock cycles](Diagram.png)

- e.g., out-of-order execution is used to avoid “pipeline stalls” that delay instruction execution.

An Overview of Sequential Programming

• Sequential programs have two main characteristics:
  • The textual order of statements specifies their order of execution
  • Successive statements must execute without any temporal overlap *visible* to programs

• However, lower layers in the solution stack can reorder instructions transparently

Consider the code sequence

\[
\begin{align*}
a &= b + c \\
d &= e - a
\end{align*}
\]

Assuming \( a, b, c, d, \) & \( e \) are in memory & loads/stores take one clock cycle out-of-order, then instruction scheduling eliminates pipeline stalls

**Original code with stalls:**

```
LD    Rb,  b
LD    Rc,  c

stall

ADD   Ra,  Rb,  Rc
SD    Ra,  a
LD    Re,  e

stall

SUB   Rd,  Re,  Ra
SD    Rd,  d
```
An Overview of Sequential Programming

• Sequential programs have two main characteristics:
  • The textual order of statements specifies their order of execution.
  • Successive statements must execute without any temporal overlap visible to programs.
  • However, lower layers in the solution stack can reorder instructions transparently.

Consider the code sequence:

\[
\begin{align*}
a &= b + c \\
d &= e - a
\end{align*}
\]

Assuming \(a, b, c, d, \) & \(e\) are in memory & loads/stores take one clock cycle out-of-order, then instruction scheduling eliminates pipeline stalls.

Original code with stalls:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Original Code</th>
<th>Scheduled Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD Rb, b</td>
<td></td>
<td>LD Rb, b</td>
</tr>
<tr>
<td>LD Rc, c</td>
<td></td>
<td>LD Rc, c</td>
</tr>
<tr>
<td>ADD Ra, Rb, Rc</td>
<td></td>
<td>ADD Ra, Rb, Rc</td>
</tr>
<tr>
<td>SD Ra, a</td>
<td></td>
<td>SD Ra, a</td>
</tr>
<tr>
<td>LD Re, e</td>
<td></td>
<td>LD Re, e</td>
</tr>
<tr>
<td>SUB Rd, Re, Ra</td>
<td></td>
<td>SUB Rd, Re, Ra</td>
</tr>
<tr>
<td>SD Rd, d</td>
<td></td>
<td>SD Rd, d</td>
</tr>
</tbody>
</table>

Scheduled code without stalls:

See en.wikipedia.org/wiki/Instruction_scheduling
An Overview of Sequential Programming

- Sequential programs have two main characteristics:
  - The textual order of statements specifies their order of execution
  - Successive statements must execute without any temporal overlap visible to programs
- However, lower layers in the solution stack can reorder instructions transparently
  - Mercifully these optimizations occur “under the hood”!!

Consider the code sequence
\[
\begin{align*}
  a &= b + c \\
  d &= e - a
\end{align*}
\]

End of Overview of Sequential Programming Concepts