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*
Learning Objectives in this Lesson

- Understand the meaning of key concepts associated with sequential programming
- Recognize the pros & cons of sequential programming
Learning Objectives in this Lesson

- Understand the meaning of key concepts associated with sequential programming
- Recognize the pros & cons of sequential programming

Overcoming these ‘cons’ motivates our upcoming focus on concurrent & parallel programming techniques for the Java & Android platforms.
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

See en.wikipedia.org/wiki/Deterministic_algorithm
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](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 characteristics

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

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

See `src/share/classes/java/util/ArrayList.java`

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

See [en.wikipedia.org/wiki/Solution_stack](en.wikipedia.org/wiki/Solution_stack)
An Overview of Sequential Programming

- Sequential programs have two 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.

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

Sequential programs have two 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:

\[
\begin{align*}
LD & \quad Rb, b \\
LD & \quad Rc, c & \text{stall} & \quad \text{stall} \\
ADD & \quad Ra, Rb, Rc \\
SD & \quad Ra, a \\
LD & \quad Re, e & \text{stall} & \quad \text{stall} \\
SUB & \quad Rd, Re, Ra \\
SD & \quad Rd, d
\end{align*}
\]
An Overview of Sequential Programming

- Sequential programs have two 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
- ADD Ra, Rb, Rc
- SD Ra, a
- LD Re, e
  - stall
- SUB Rd, Re, Ra
- SD Rd, d

Scheduled code without stalls:

- LD Rb, b
- LD Rc, c
- ADD Ra, Rb, Rc
- SD Ra, a
- LD Re, e
- SUB Rd, Re, Ra
- SD Rd, d

See en.wikipedia.org/wiki/Instruction_scheduling
Evaluating the Pros & Cons of Sequential Programming
Evaluating the Pros & Cons of Sequential Programming

• Pros of sequential programming
Evaluating the Pros & Cons of Sequential Programming

- Pros of sequential programming
  - Easy to program & debug
Evaluating the Pros & Cons of Sequential Programming

• Pros of sequential programming
  • Easy to program & debug
  • “Intuitive” since it matches the steps expressed in algorithms

```c
int i, j, len = ...;
for (i = 0; i < len - 1; i++) {
    int min = i;
    for (j = i + 1; j < len; j++)
        if (a[j] < a[min])
            min = j;
    if (min != i)
        swap(a[i], a[min]);
}
```

See [en.wikipedia.org/wiki/Selection_sort](http://en.wikipedia.org/wiki/Selection_sort)
This algorithm can be understood by reading it as written, i.e., there are no “surprises”

Evaluating the Pros & Cons of Sequential Programming

- Pros of sequential programming
  - Easy to program & debug
    - “Intuitive” since it matches the steps expressed in algorithms
  - The behavior in the debugger reflects actual program behavior
Evaluating the Pros & Cons of Sequential Programming

• Pros of sequential programming
  • Easy to program & debug
    • “Intuitive” since it matches the steps expressed in algorithms
  • The behavior in the debugger reflects actual program behavior
  • Conversely, the behavior of non-sequential programs often differ when run in a debugger vs. “in the wild”

These differences stem from perturbations in timing from the different execution contexts
Evaluating the Pros & Cons of Sequential Programming

- Pros of sequential programming
  - Easy to program & debug
  - Deterministic execution order simplifies reasoning about & assuring program behavior

See screenprism.com/insights/article/what-is-the-ludovico-technique-and-how-does-it-work
Evaluating the Pros & Cons of Sequential Programming

• Pros of sequential programming
  • Easy to program & debug
  • Deterministic execution order simplifies reasoning about & assuring program behavior
  • Especially for safety-critical cyber-physical systems

See en.wikipedia.org/wiki/Cyber-physical_system
Evaluating the Pros & Cons of Sequential Programming

• Pros of sequential programming
  • Easy to program & debug
  • Deterministic execution order simplifies reasoning about & assuring program behavior
  • Especially for safety-critical cyber-physical systems

The right answer delivered too late becomes the wrong answer
Evaluating the Pros & Cons of Sequential Programming

- Cons of sequential programming
Evaluating the Pros & Cons of Sequential Programming

- Cons of sequential programming
  - Cannot leverage the parallelism available in multi-core systems

See [en.wikipedia.org/wiki/Multi-core_processor](en.wikipedia.org/wiki/Multi-core_processor)
Evaluating the Pros & Cons of Sequential Programming

• Cons of sequential programming
  • Cannot leverage the parallelism available in multi-core systems
  • Performance may therefore suffer relative to concurrent & parallel programs
Evaluating the Pros & Cons of Sequential Programming

- Cons of sequential programming
  - Cannot leverage the parallelism available in multi-core systems
  - It’s hard to be responsive to multiple I/O sources/sinks

  e.g., mouse movement/clicks, touch events, GPS location signals, network connections, asynchronous storage read & write completions, etc.

See en.wikipedia.org/wiki/Responsiveness
Evaluating the Pros & Cons of Sequential Programming

- Cons of sequential programming
  - Cannot leverage the parallelism available in multi-core systems
  - It’s hard to be responsive to multiple I/O sources/sinks

Having only a single thread of control complicates the structure of sequential programs for blocking operations

Evaluating the Pros & Cons of Sequential Programming

• Cons of sequential programming
  • Cannot leverage the parallelism available in multi-core systems
  • It’s hard to be responsive to multiple I/O sources/sinks

Overcoming these ‘cons’ motivates all of the concurrency & parallelism topics that we cover henceforth!!!
End of Overview of Sequential Programming Concepts