Overview of Sequential Programming Concepts

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

Professor of Computer Science
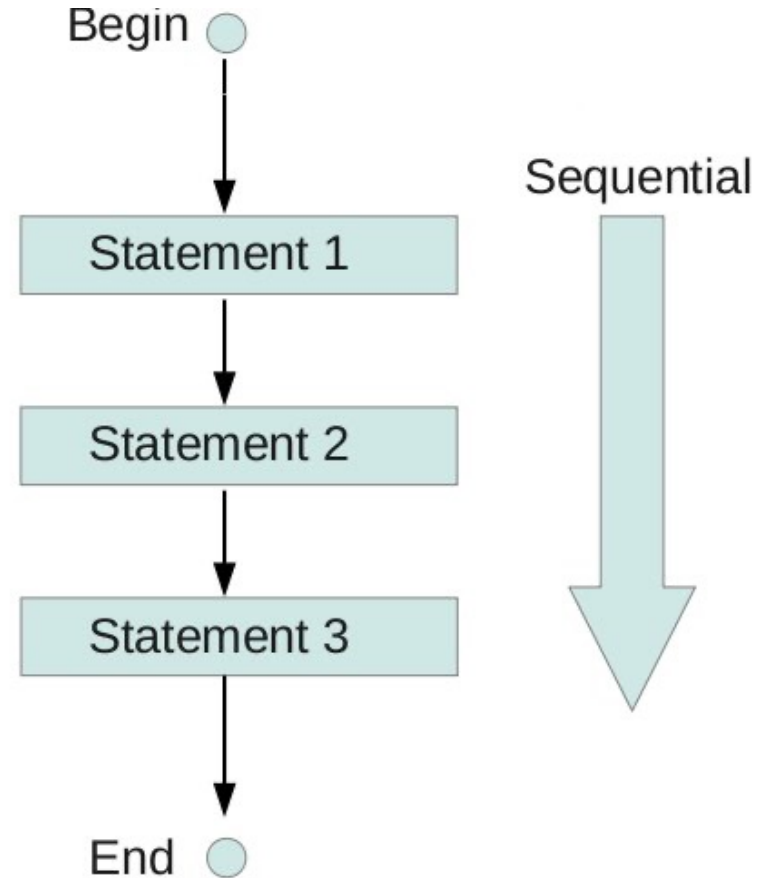
Institute for Software
Integrated Systems

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
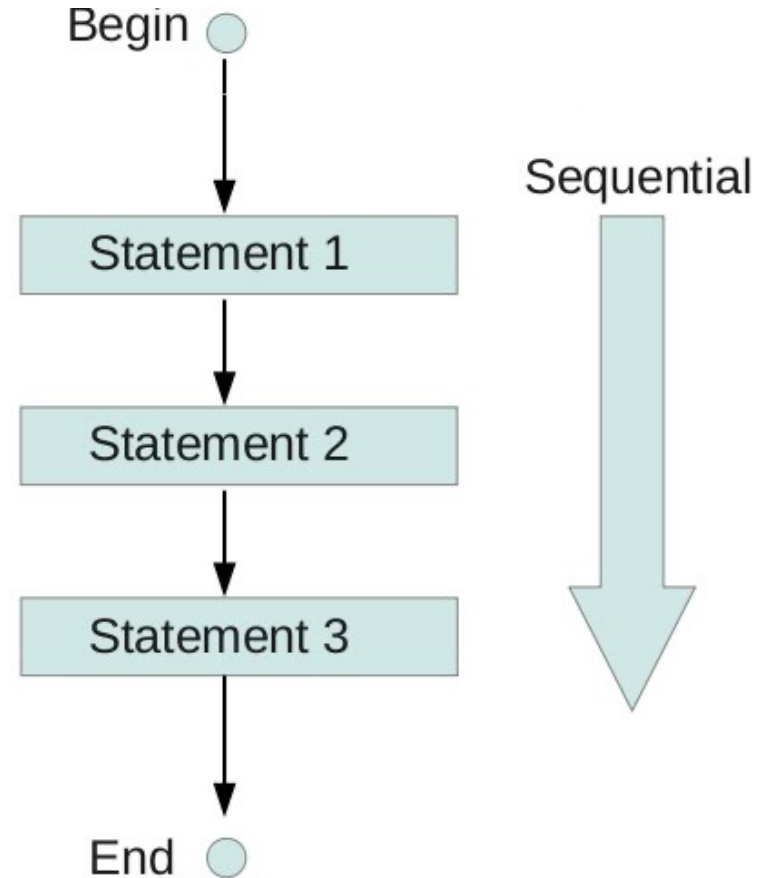
Learning Objectives in this Lesson

- 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*

Begin ○

↓

Statement 1

↓

Statement 2

↓

Statement 3

↓

End ○

Sequential ⇩

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*
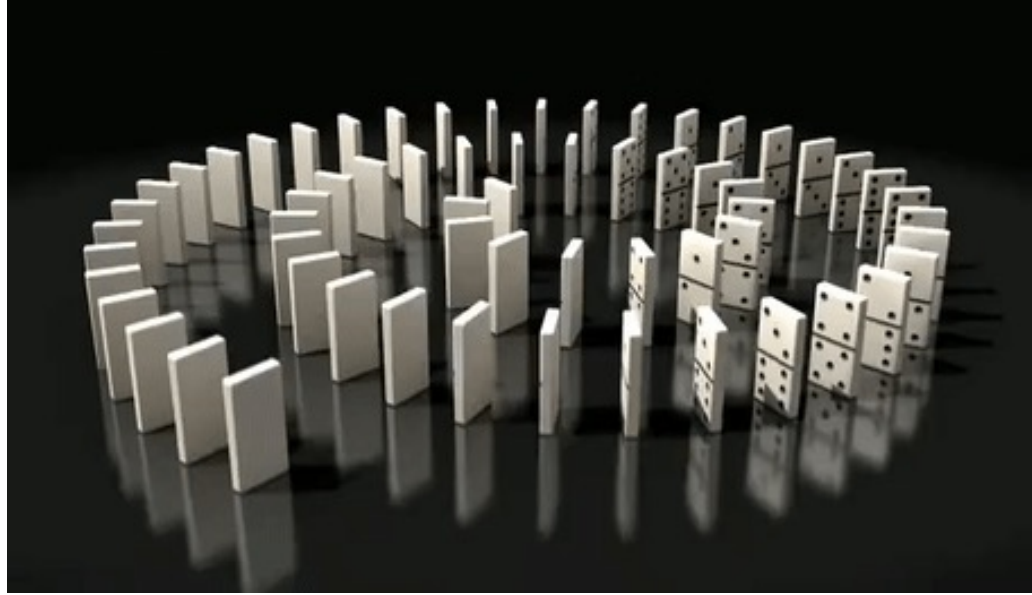
See en.wikipedia.org/wiki/Sequential_algorithm

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
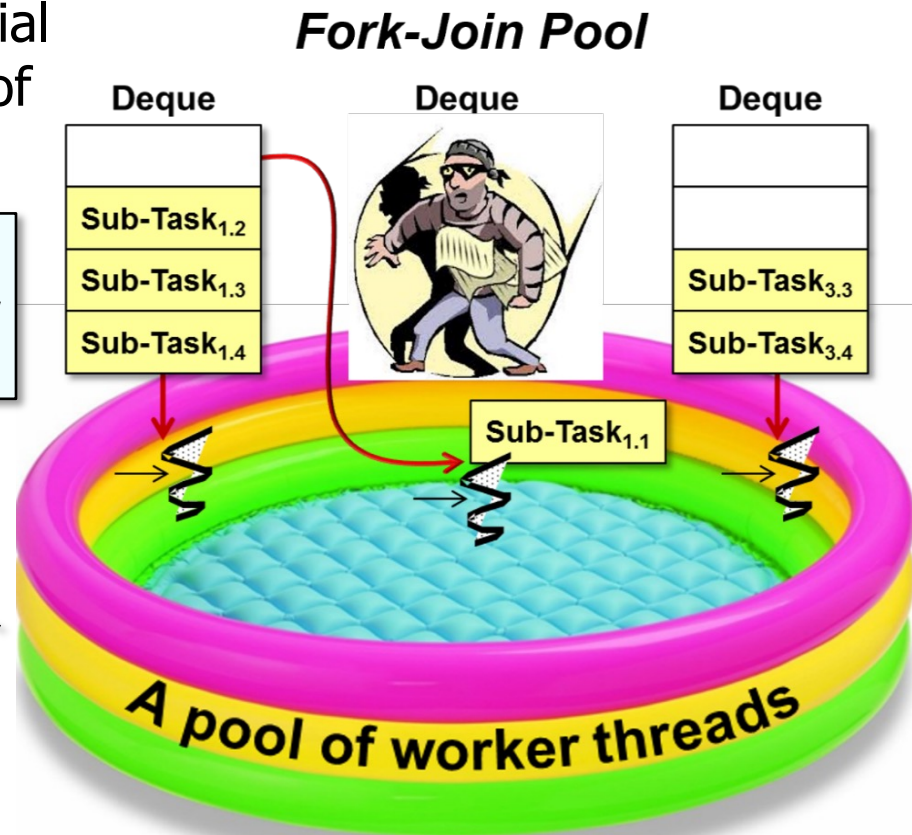
See en.wikipedia.org/wiki/Randomized_algorithm

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

Fork-Join Pool

Deque	Deque	Deque
Sub-Task$_{1.2}$		
Sub-Task$_{1.3}$		Sub-Task$_{3.3}$
Sub-Task$_{1.4}$		Sub-Task$_{3.4}$

Sub-Task$_{1.1}$

A pool of worker threads

See gee.cs.oswego.edu/dl/papers/fj.pdf

An Overview of Sequential Programming

- Sequential programs have two main characteristics
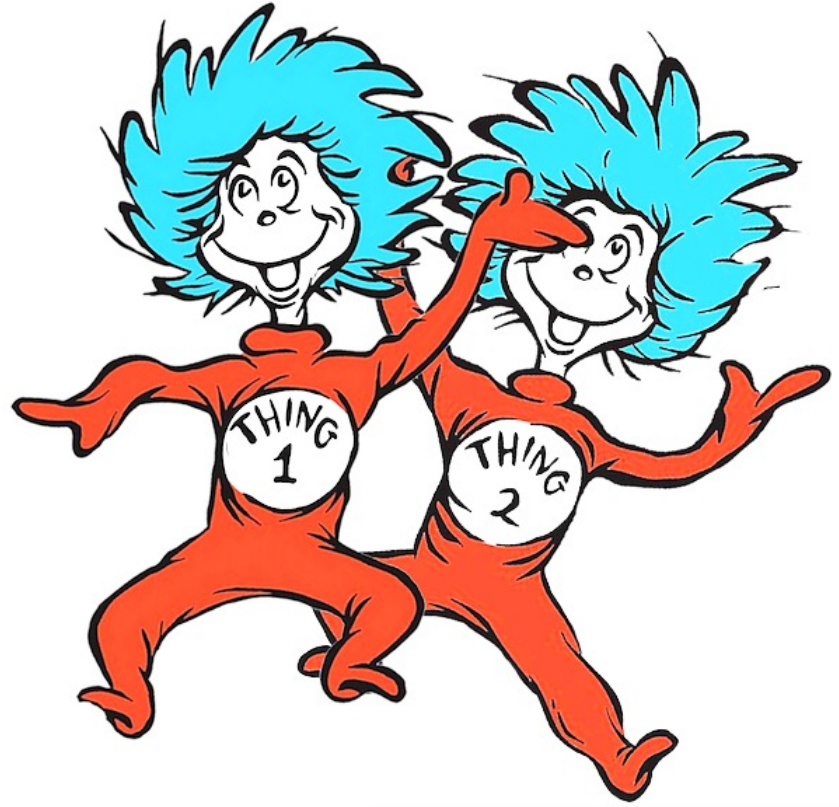
An Overview of Sequential Programming

- Sequential programs have two main characteristics:

  - The textual order of statements specifies their order of execution

```
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

```
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()!!!

See src/share/classes/java/util/ArrayList.java
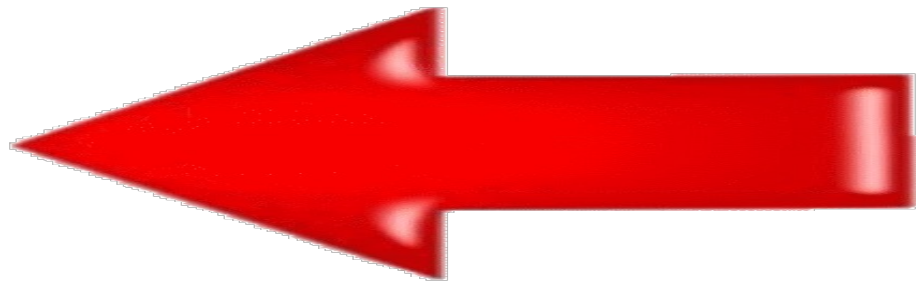
An Overview of Sequential Programming

- 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

    $a = b + c$

    $d = e - a$
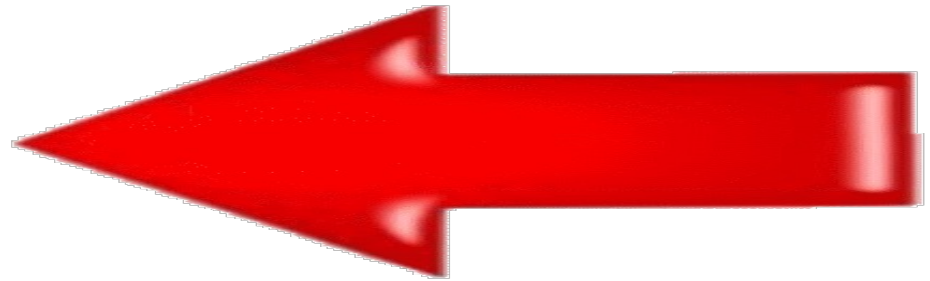
An Overview of Sequential Programming

- 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

$a = b + c$

$d = e - a$

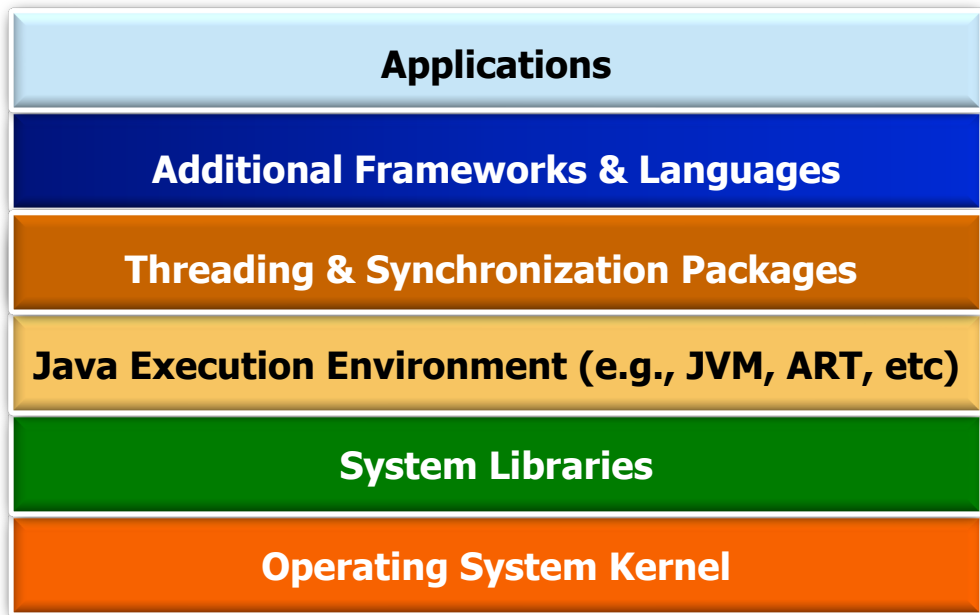> The value of 'a' must be assigned before the value of 'd' is assigned

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 "stack" layers can reorder instructions invisibly to programs

OUT OF ORDER

Consider the code sequence

$$a = b + c$$
$$d = e - a$$

Applications
Additional Frameworks & Languages
Threading & Synchronization Packages
Java Execution Environment (e.g., JVM, ART, etc)
System Libraries
Operating System Kernel

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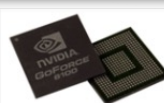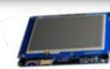
An Overview of Sequential Programming

- 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 "stack" layers can reorder instructions invisibly to programs

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

Consider the code sequence

a = b + c

d = e - a

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

See en.wikipedia.org/wiki/Out-of-order_execution & en.wikipedia.org/wiki/Pipeline_stall

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

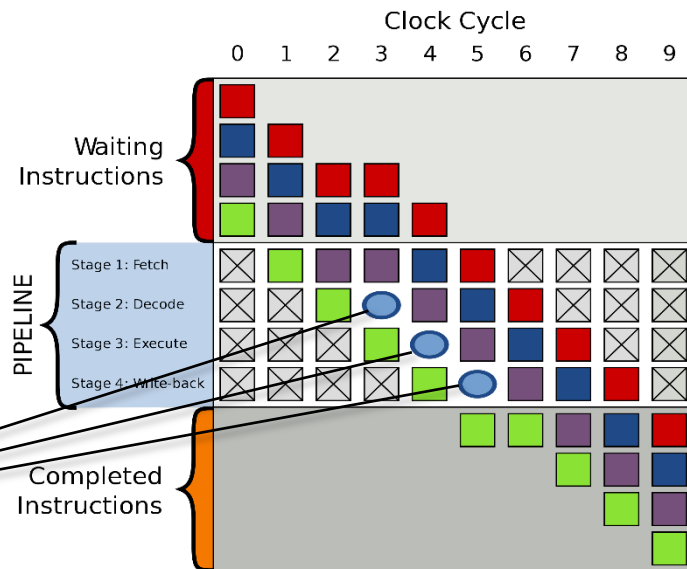   - However, lower "stack" layers can reorder instructions invisibly to programs

Consider the code sequence

> a = b + c
>
> d = e - a

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 "stack" layers can reorder instructions invisibly to programs

Consider the code sequence

$$a = b + c$$
$$d = e - a$$

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
```

Scheduled code without stalls:

```
LD    Rb, b
LD    Rc, c
LD    Re, e
ADD   Ra, Rb, Rc
SD    Ra, a
SUB   Rd, Re, Ra
SD    Rd, d
```
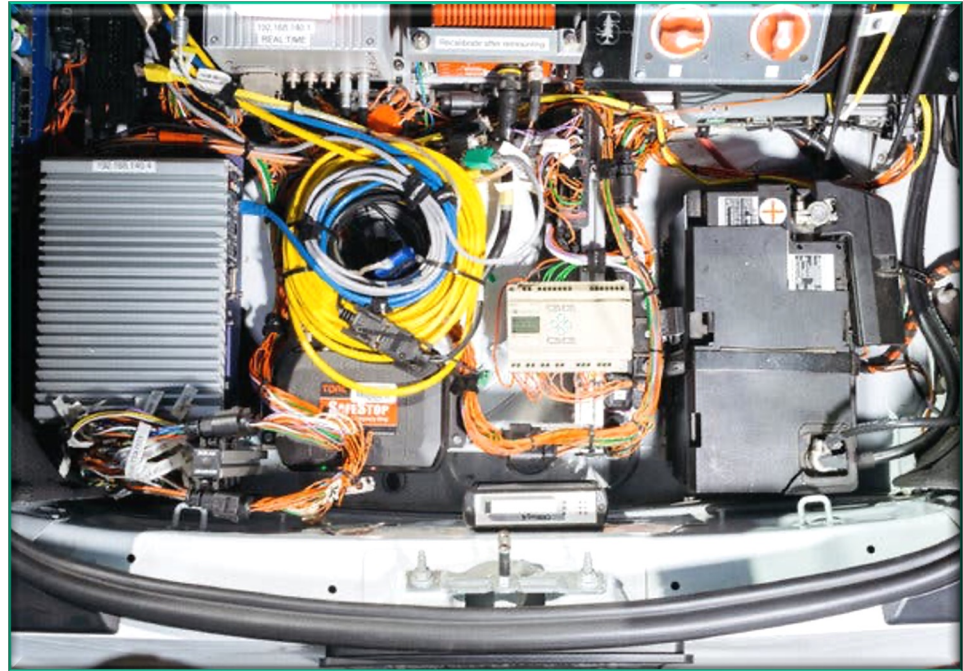
See en.wikipedia.org/wiki/Instruction_scheduling

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 "stack" layers can reorder instructions invisibly to programs

      - Mercifully, such optimizations occur "under the hood"!!

Consider the code sequence
$$a = b + c$$
$$d = e - a$$

See www.nytimes.com/2015/09/27/business/complex-car-software-becomes-the-weak-spot-under-the-hood.html

End of Overview of Sequential Programming Concepts