Real Time and Embedded Systems

by

Dr. Lesley Shannon

Email: lshannon@ensc.sfu.ca

Course Website: http://www.ensc.sfu.ca/~lshannon/courses/ensc351

Simon Fraser University

Slide Set: 1

Date: September 8, 2011

Slide Set Overview

- Processor Architecture Review
- What is single threaded vs multi-thread processing?
  - An analogy
  - What about SMT vs CMP/SMP processors?
- What are processes vs threads?
- How do we draw/represent multi-threaded software programs?
  - An introduction to Collaboration Graph Notation

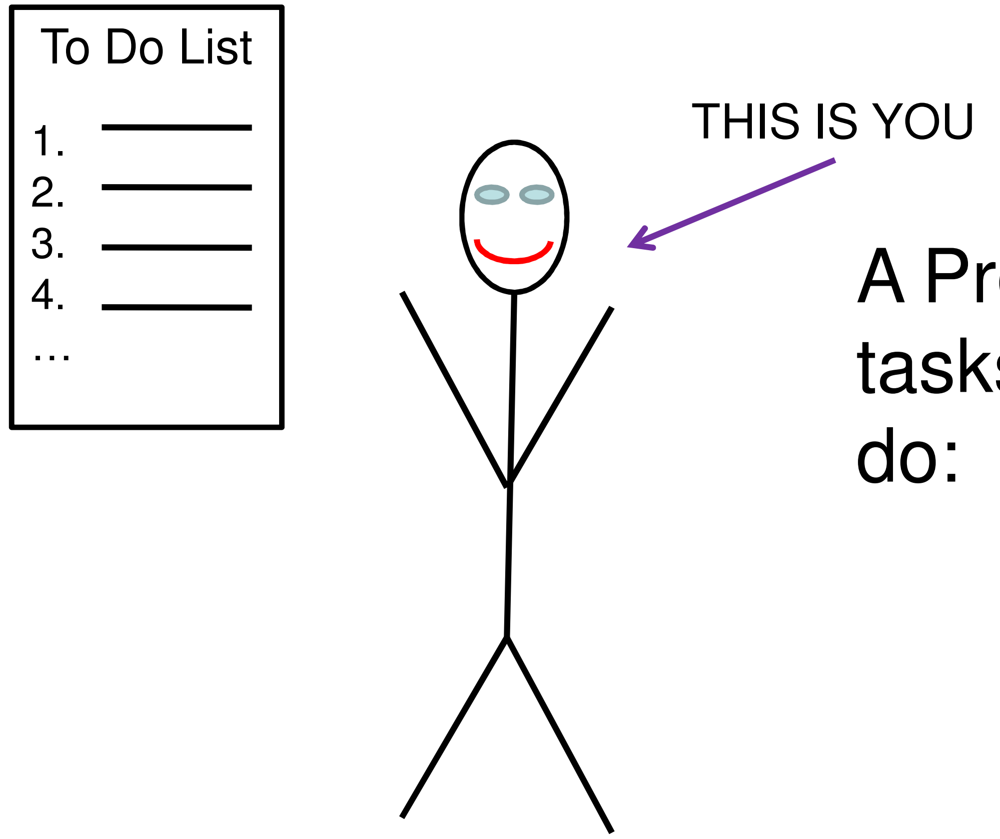
Processor Architecture Review

Why have this review

- We're going to be talking about multi-threaded programming & interrupts
  - To understand how the software works and some of the challenges, you need to understand the platform

- Understanding how a processor (& compiler) works leads to better coding practices
  - Analogy: A mechanic has a better understanding of a car than a driver

Analogy of processor operation

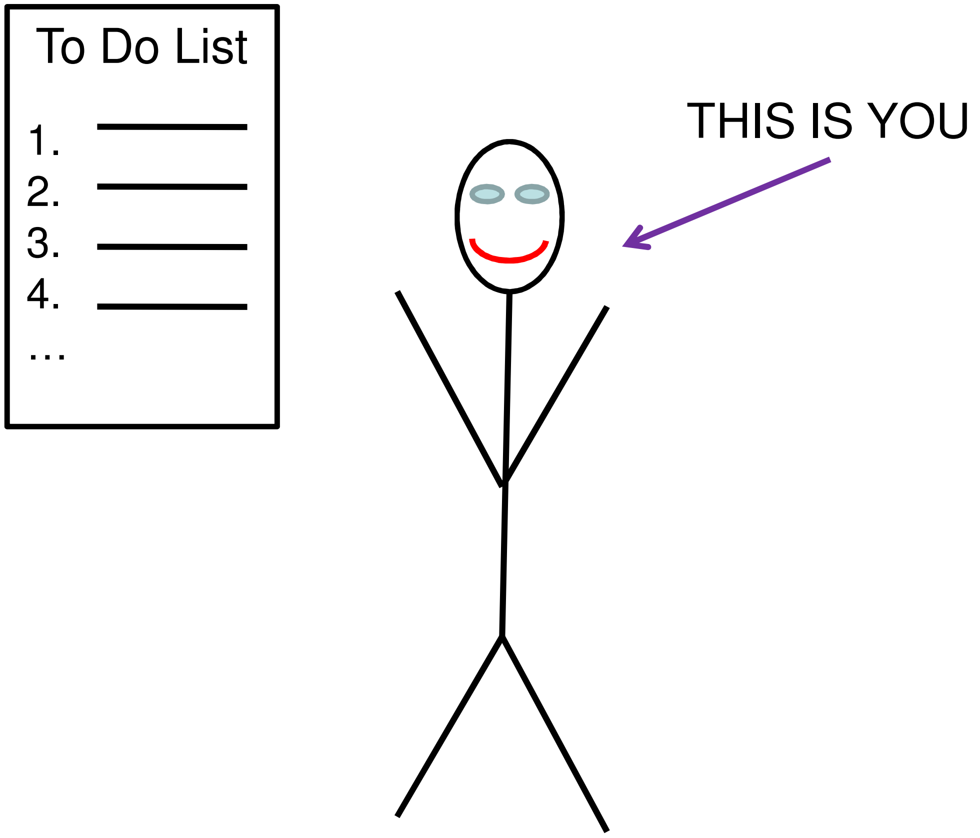


A Processor accomplishes tasks in the same way you do:

-Sequentially

How do you go about accomplishing a “To Do” List?

Analogy of processor operation



How do you go about accomplishing a “To Do” List?

How does a processor operate?

Recall how you accomplish your “To Do” List.

Processor Architecture (draw here)

Processor Optimizations

1.

2.

3.

4.

5.

Processor Architecture (draw here)

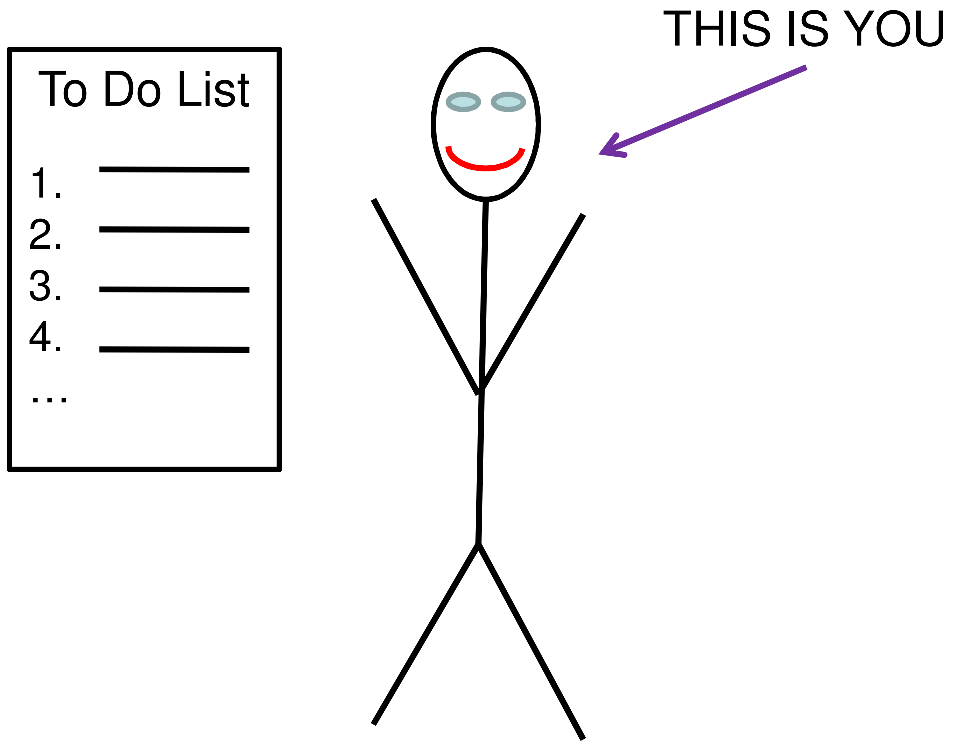
Summary

- Processors are sequential machines
  - Multi-threaded applications built on this platform will have to play neat tricks to share the resources and meet their deadlines
  - The Operating System will provide a level of abstraction between the low-level architecture and the multi-threaded application you want to run

What is single threaded vs multi-threaded processing?

Single vs Multi-threaded Programming

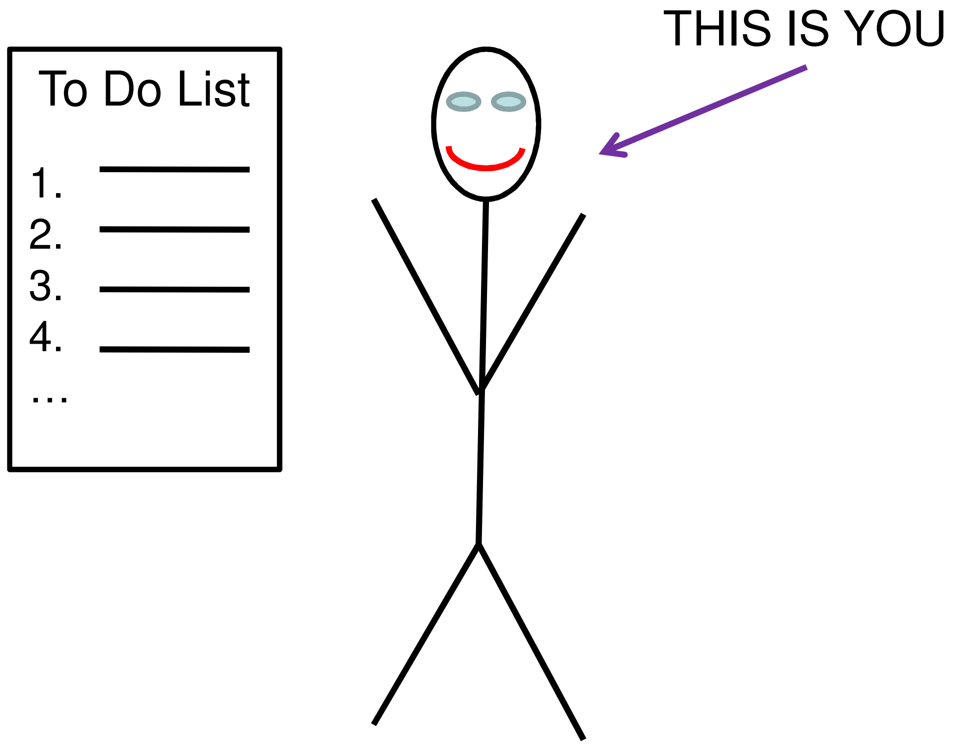
- Let’s recall our analogy from last Thursday of you with your “To Do” list:
  - Bake a 6 course meal



Single vs Multi-threaded Programming

- Let’s recall our analogy from last Thursday of you with your “To Do” list:
  - Bake a 6 course meal

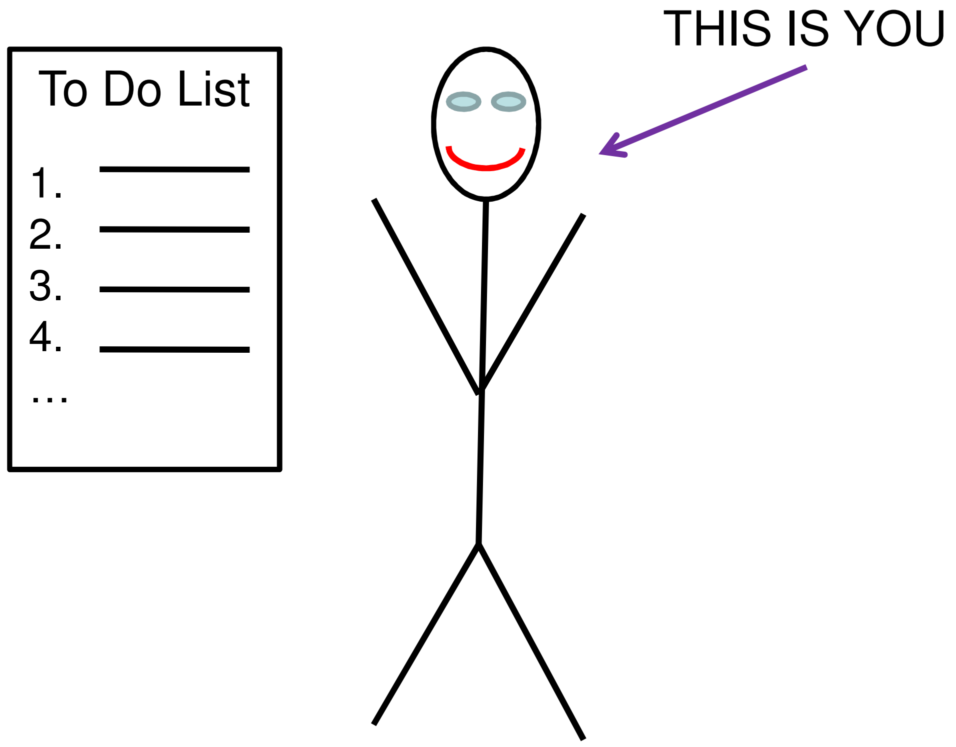
- Performing your “To Do” list is analogous to a thread executing a piece of code



Single vs Multi-threaded Programming

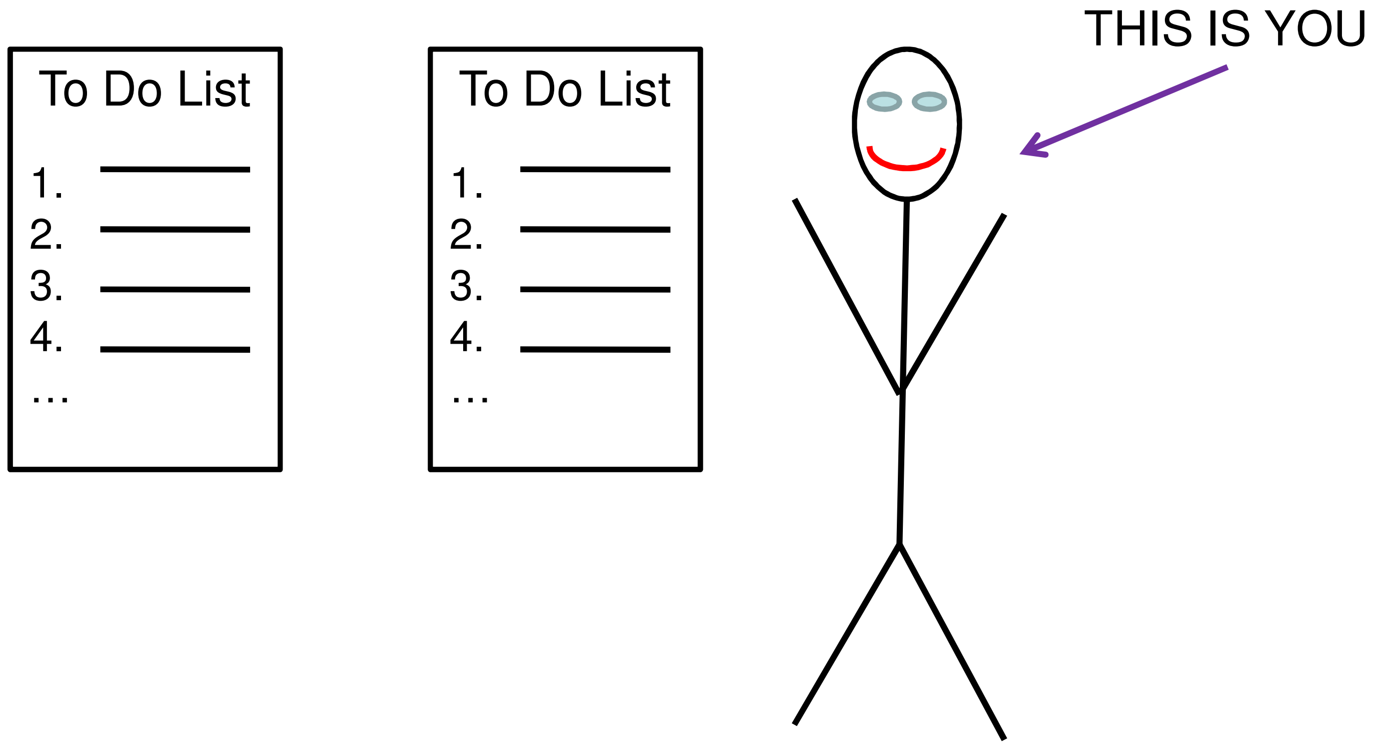
- Let’s recall our analogy from last Thursday of you with your “To Do” list:
  - Bake a 6 course meal

- You are the processor executing the single thread



Single vs Multi-threaded Programming

- But what if you have ***2*** "To Do" lists?
  - Bake a 6 course meal ***AND***
  - Clean your home



How would you go about completing these 2 tasks?

- Bake a 6 course meal
- Clean your home

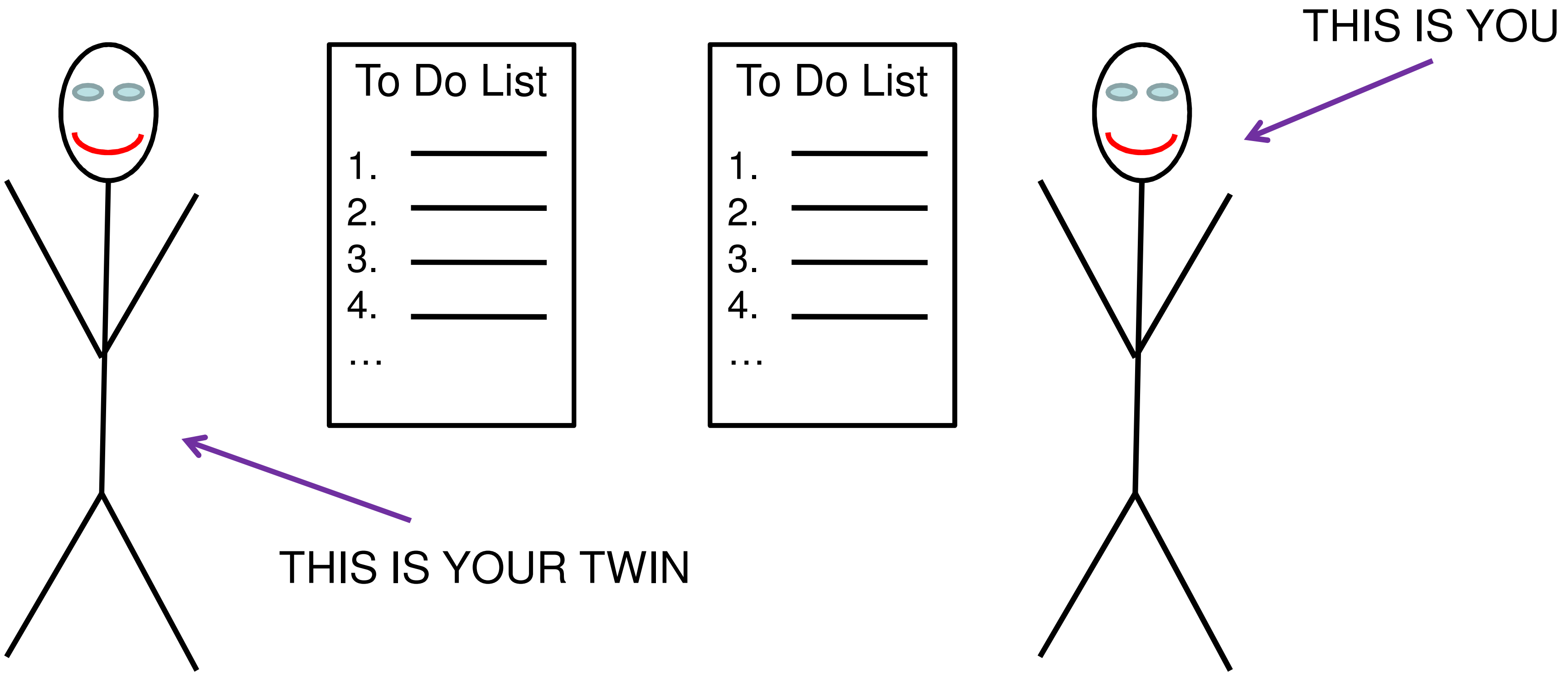
1.

2.

3.

Single vs Multi-threaded Programming

- What if you have 2 “To Do” lists and your twin helps out?
  - Bake a meal
  - Clean your home



How would you go about completing these 2 tasks?

- Bake a 6 course meal
- Clean your home

1.

2.

3.

4.

Summary: Single vs Multi-threaded Programming

- Single threaded

- Multi-threaded

Summary: SMT vs CMP

- SMT processors

- CMP/SMP

What are processes vs threads?

Processes

- Processes include:
  - An address space
    - Defines the associated code and data pages
  - OS resources
    - E.g. open files
  - Accounting information
    - CPU time
    - Memory
    - Etc

Processes

- Processes include:
  - Execution State
    - PC
    - SP
    - Register File
    - ??

Processes

- Creating a process requires:
  - The allocation and initialization of data structures
- Processes run on top of the OS
  - Why is this a good thing?
    - Hint: Recall Last class

Processes

- To communicate between processes
  - Normally through the OS
    - Incurs the overhead of system calls and copying data
- What about threads?

Threads

- They share:
  - The same address space
    - Code and data
  - The same privileges
  - The same resources
    - Files
    - Sockets
    - Etc
- Because they are part of the same process!

Threads

- What is a thread?
  - The execution state of a process
    - In other words, the "***<u>thread</u>***" of control
  - The execution state includes:
    - The PC
    - The SP
    - The Register File
    - ??

Summary of Processes vs Threads (General)

- A thread is a sequential execution stream within a process
  - (e.g. PC, SP, registers, etc)
- A process defines the address space and general process attributes
  - Excludes thread execution

Summary of Processes vs Threads (General)

- Processes are the containers in which threads execute
  - Threads become the unit of scheduling on the processor
  - Processes are static/threads are dynamic

One last concept…

- Besides Processes and Threads, there is a "new" concept in multi-threaded programming:
  - The ***TASK***
    - Tasks are "indivisible units works"
    - They are executed by threads and …
    - Threads may comprise one or more tasks …
    - *However*, mapping individual tasks to individual threads will provide the greatest opportunities for parallel execution

Questions?

- What are the Linux function calls to create a process (check out the POSIX standards)?

- What is the Linux function call to create a thread?

Questions?

- Can a thread be bound to more than one process?

- Can a process have multiple threads?

Next Lecture: Kernel-Level vs User-Level Threads

- Kernel-LevelThreads
  - Threads that the O/S kernel “knows about”
- User-Level Threads
  - Threads within an application of which the kernel is unaware
- More next lecture …
  - Now …

Using Collaboration Graph Notation to represent multi-threaded systems

Collaboration Graph Notation

- What terminology are we familiar with thus far:
  - Process
  - Thread
- Also from programming courses:
  - Function calls
  - Object instantiation
  - Function parameters and return values
  - Libraries

Collaboration Graph Notation

Collaboration Graph Notation

Collaboration Graph Notation