

The Context Switch

Copyright ©: Nahrstedt, Angrave, Abdelzaher



Announcements/Reminders

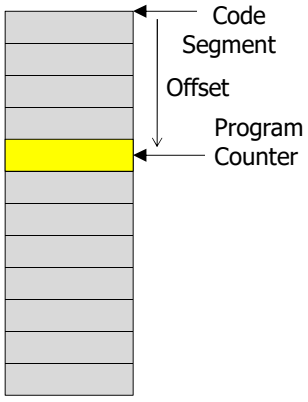
- Topics poll is open – please vote today or tomorrow
- HW1 will be out tonight. Due in a week. Please check class website. Please work on homework independently.
- Please remember to form an MP group
- Please fill out the MP group signup sheet (see post @29 on piazza) to let us know who is in your group
- TA office hours are now posted on the webpage

Implementing Multiple Threads (on One Processor)

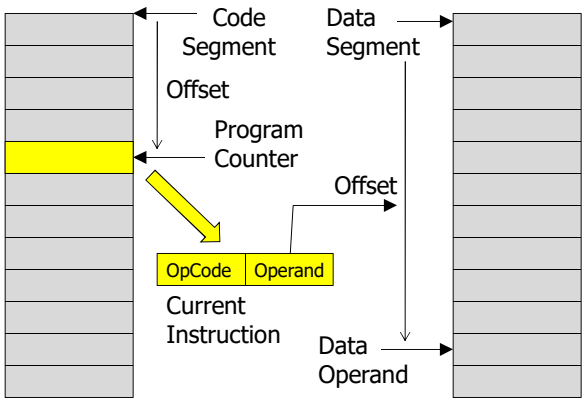
- Threads provide an illusion of concurrency
  - Given one "real CPU", give the illusion of multiple "virtual CPUs" each dedicated to a thread
  - The "virtual CPUs" transparently share the "real CPU"
- The abstraction is recursive
  - If the virtual CPU abstraction is perfect, a "virtual CPU" is indistinguishable from a "real CPU"
  - Therefore, a "virtual CPU" can play the role of a real CPU and can give the illusion of multiple (2nd-level) "virtual CPUs" transparently sharing the 1st-level virtual CPU.

3

What's a "Real CPU"?

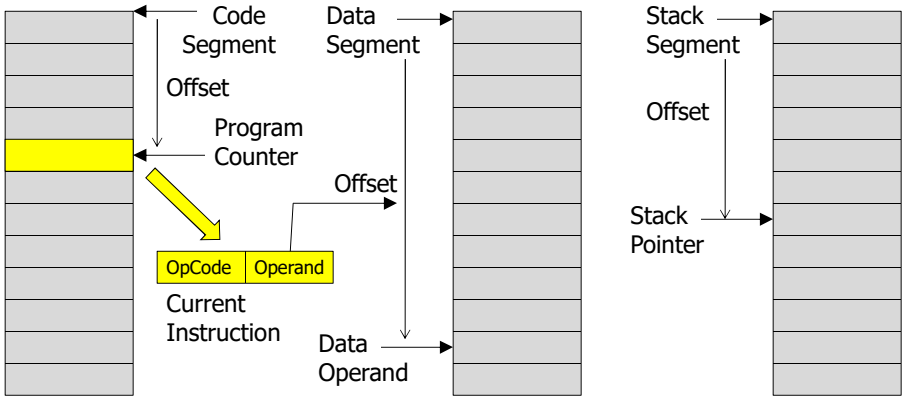


What's a "Real CPU"?



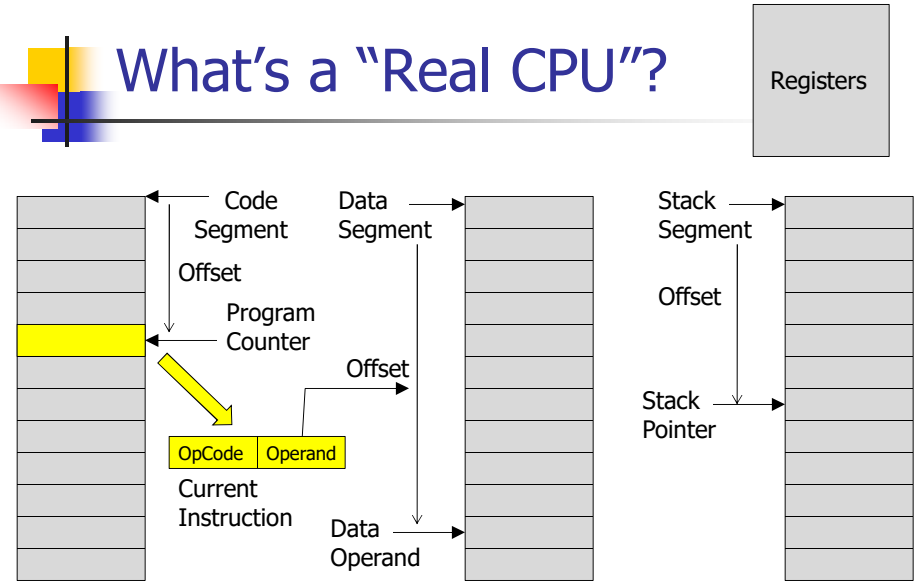
5

What's a "Real CPU"?



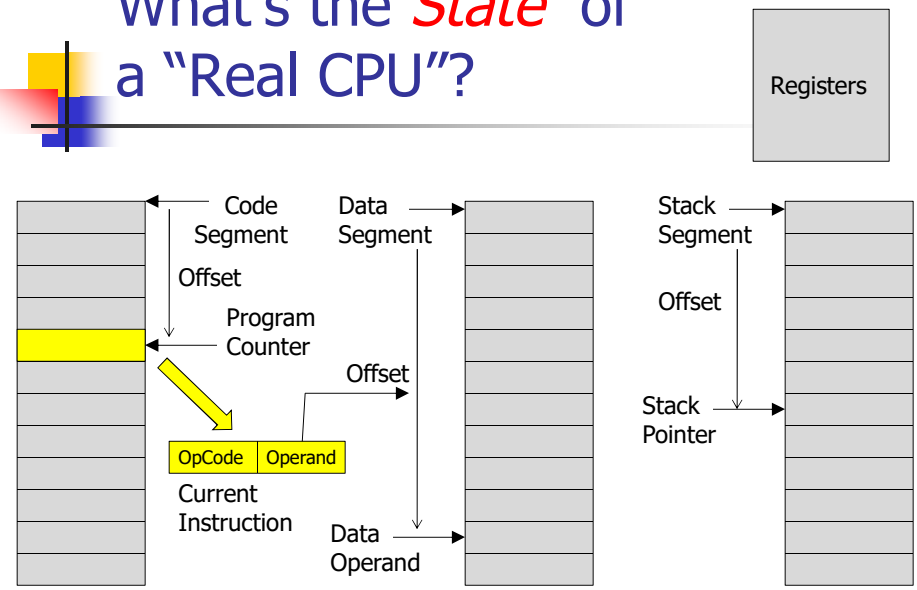
6

What's a "Real CPU"?



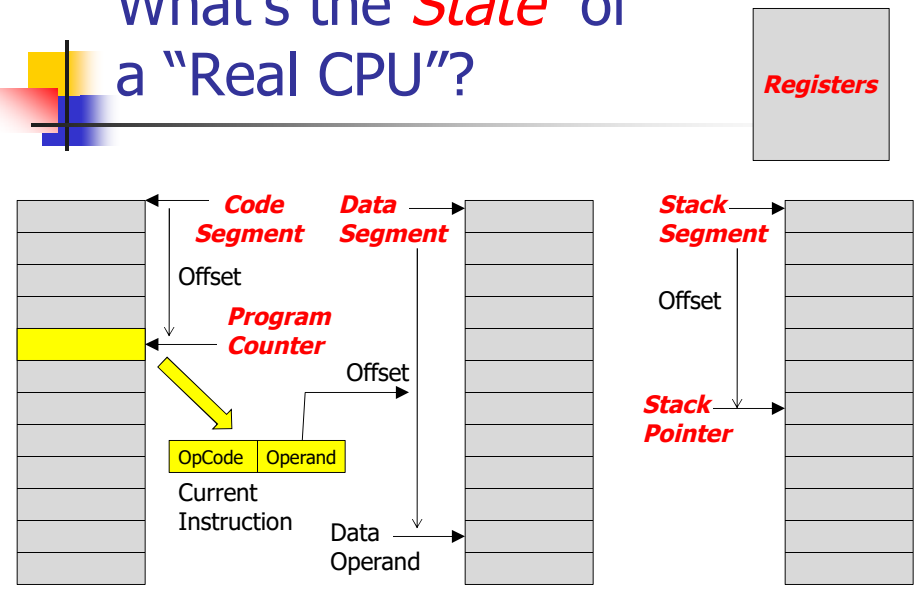
7

What's the *State* of a "Real CPU"?



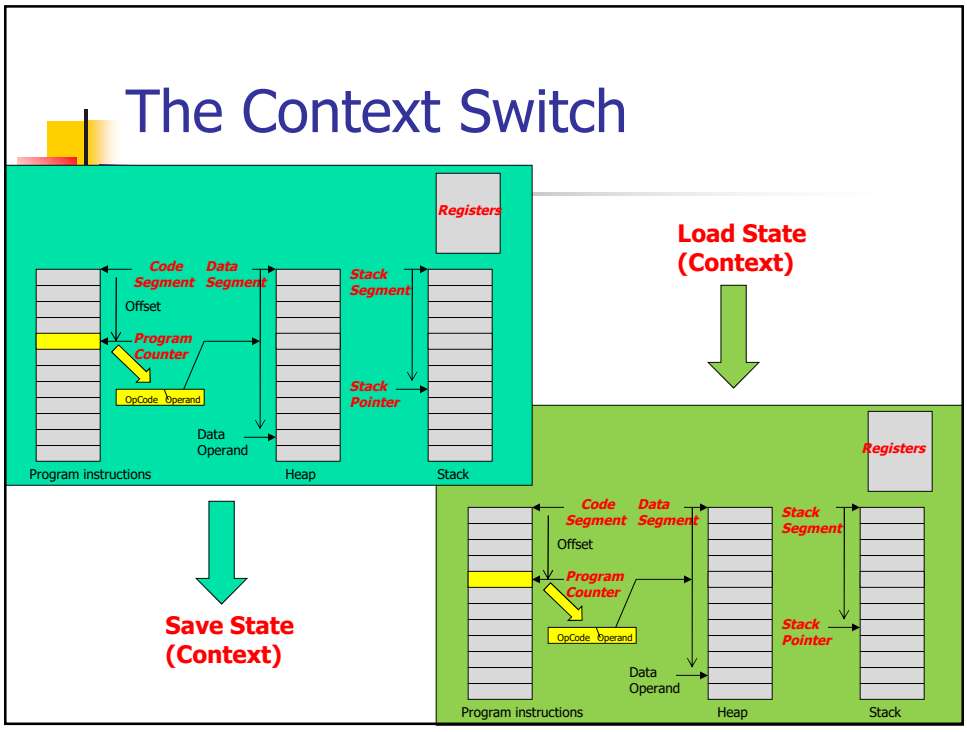
8

What's the *State* of a "Real CPU"?



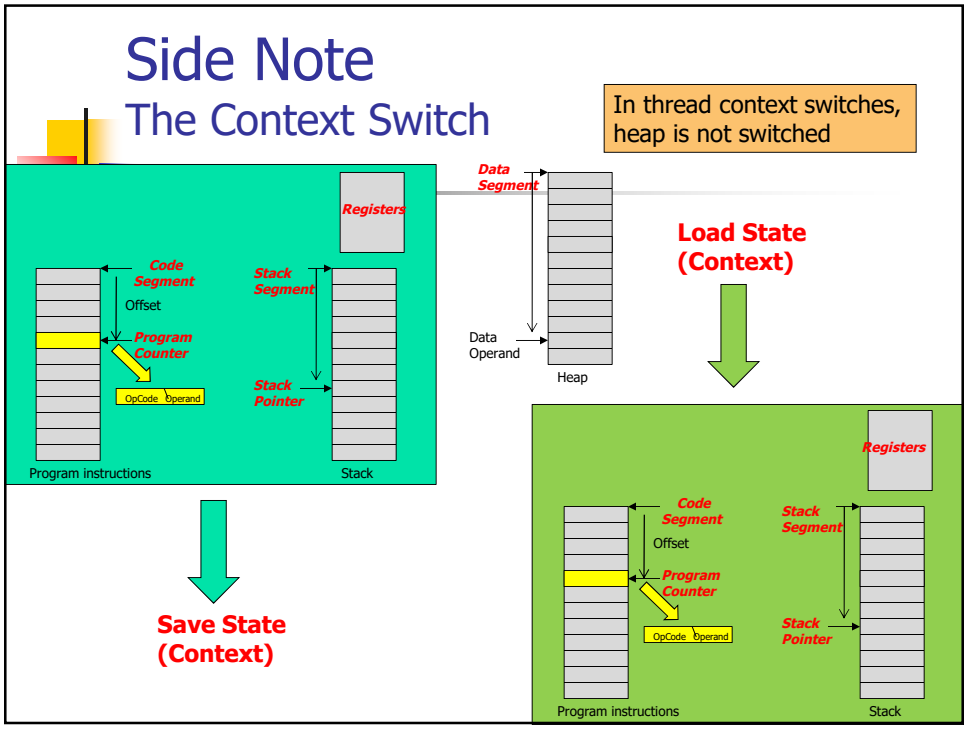
9

The Context Switch



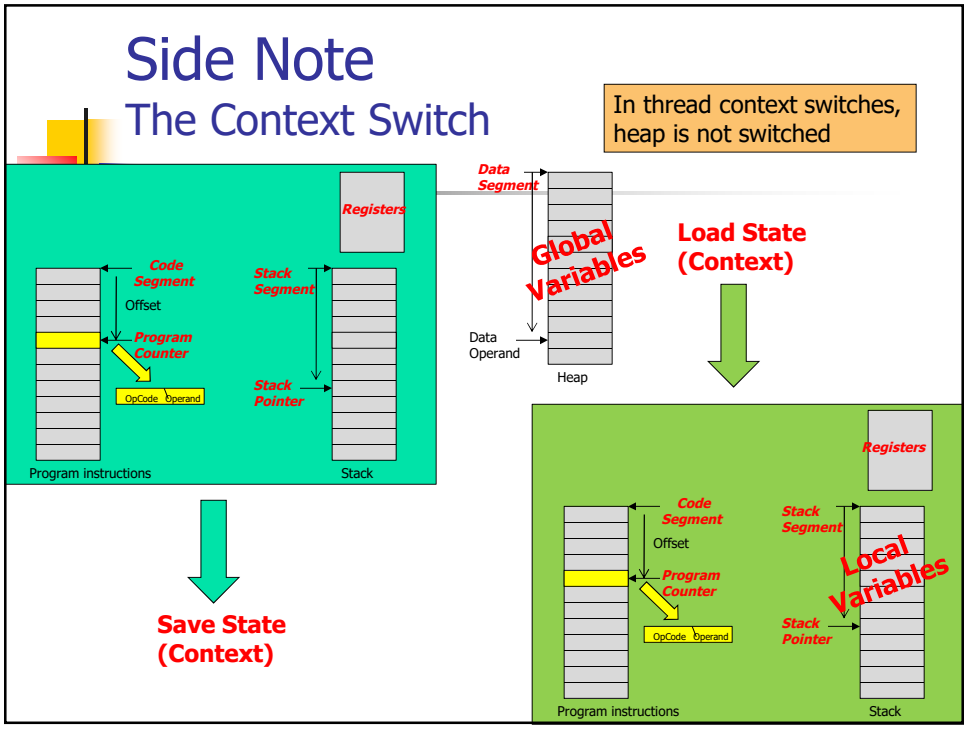
Side Note
The Context Switch

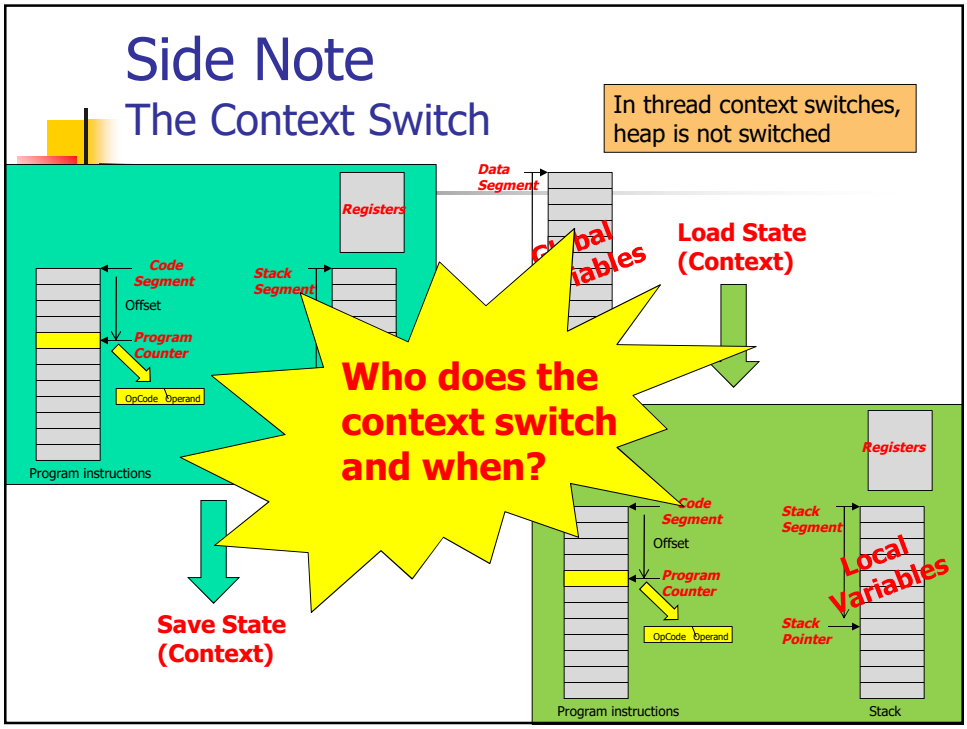
In thread context switches, heap is not switched

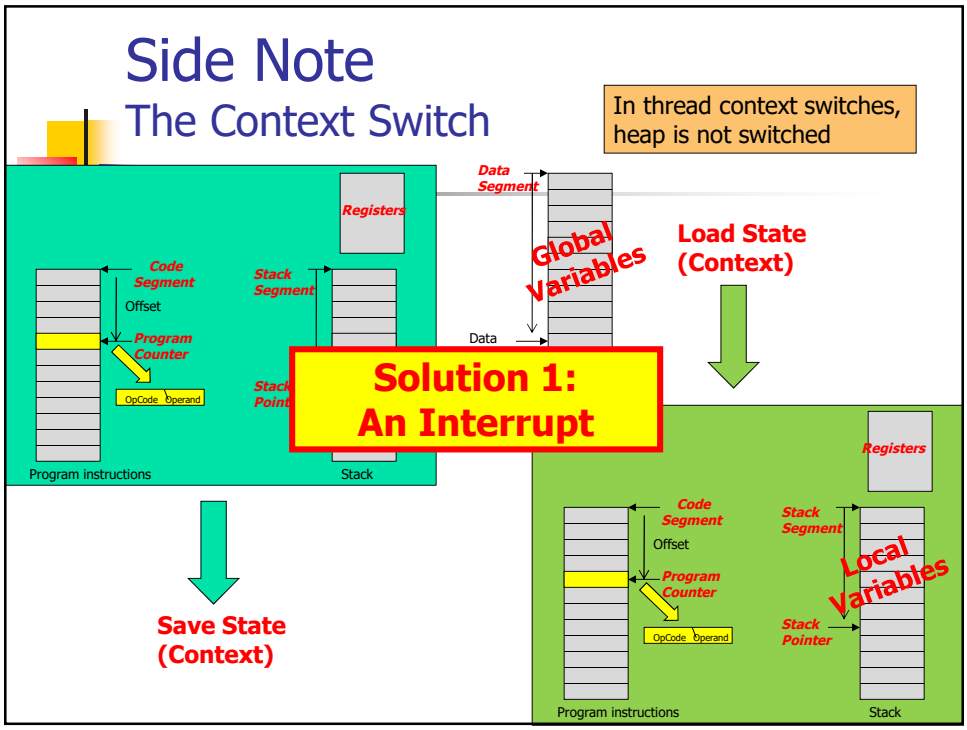


Side Note
The Context Switch

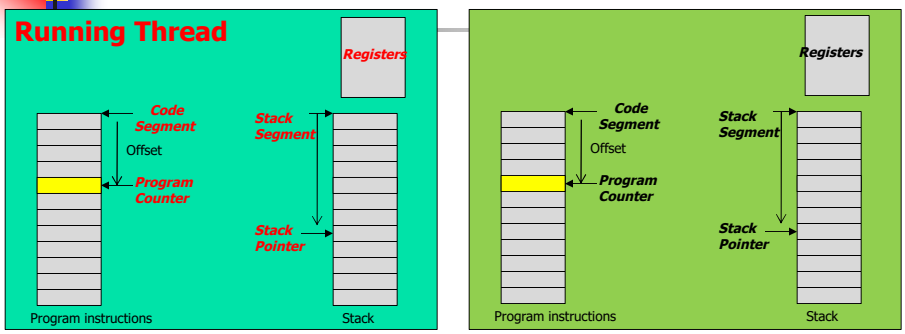
In thread context switches, heap is not switched



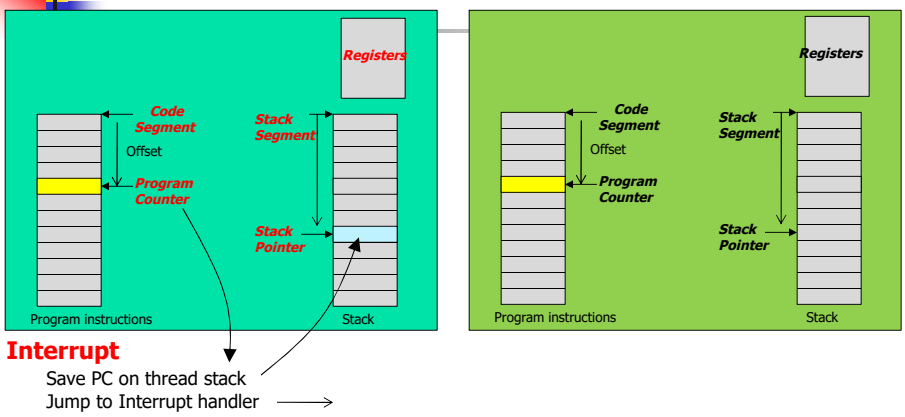

Side Note
The Context Switch
In thread context switches, heap is not switched
Data Segment
Registers
Code Segment
Stack Segment
Offset
Program Counter
OpCode
Operand
Program instructions
Load State (Context)
Who does the context switch and when?
Save State (Context)
Registers
Code Segment
Stack Segment
Offset
Program Counter
OpCode
Operand
Stack Pointer
Local Variables
Program instructions
Stack


Side Note
The Context Switch
In thread context switches, heap is not switched
Data Segment
Registers
Code Segment
Stack Segment
Offset
Program Counter
OpCode
Operand
Global Variables
Data
Load State (Context)
Solution 1: An Interrupt
Program instructions
Stack
Save State (Context)
Registers
Code Segment
Stack Segment
Offset
Program Counter
OpCode
Operand
Stack Pointer
Local Variables
Program instructions
Stack

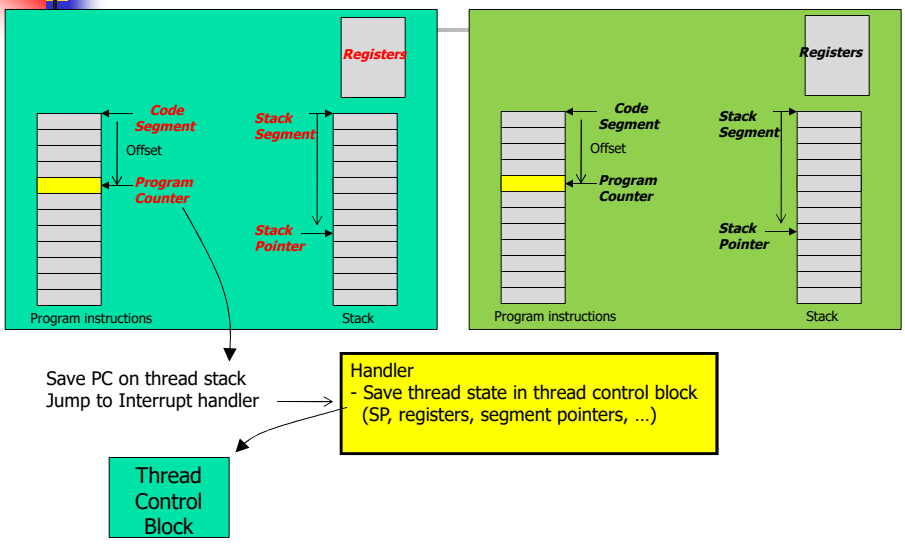
Context Switch (Interrupt)



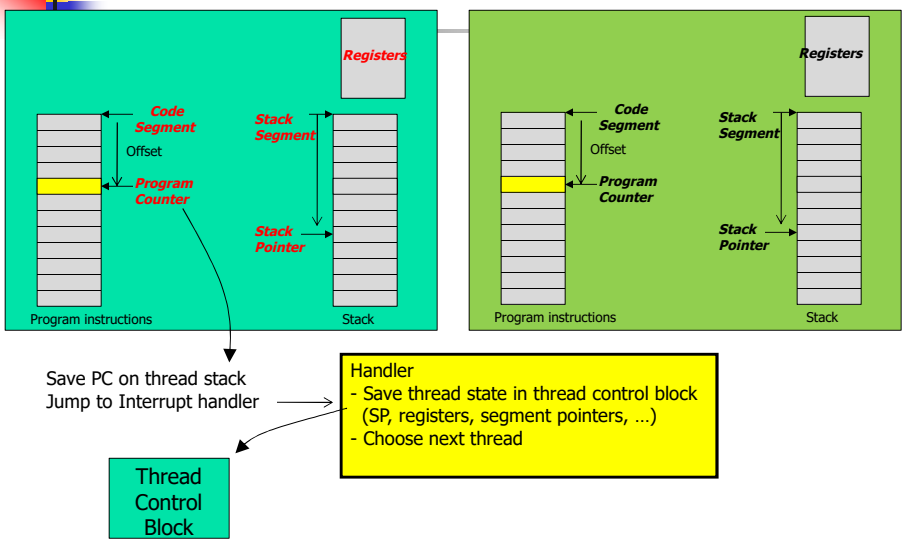
Context Switch (Interrupt)



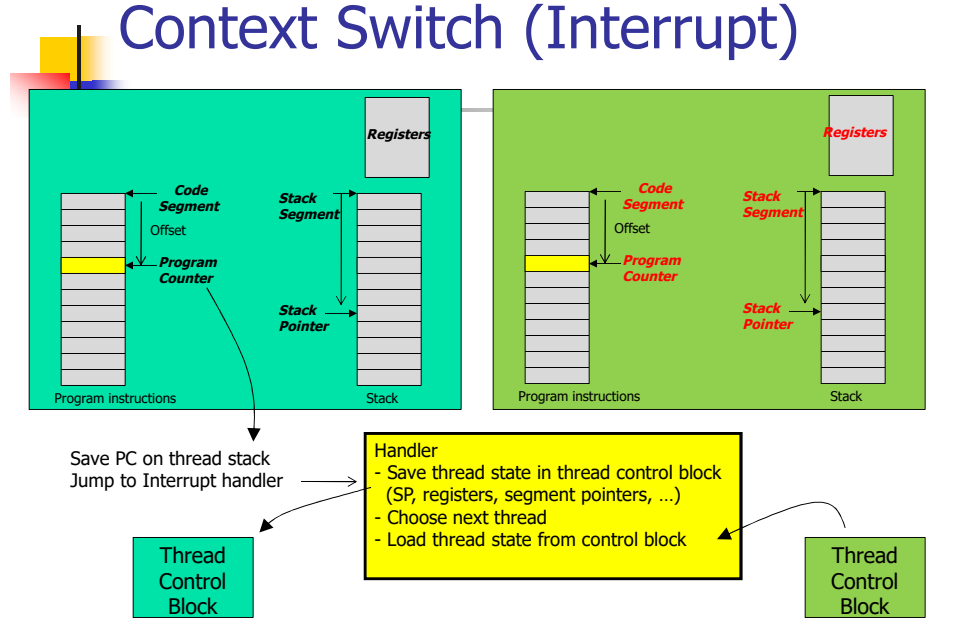
Context Switch (Interrupt)



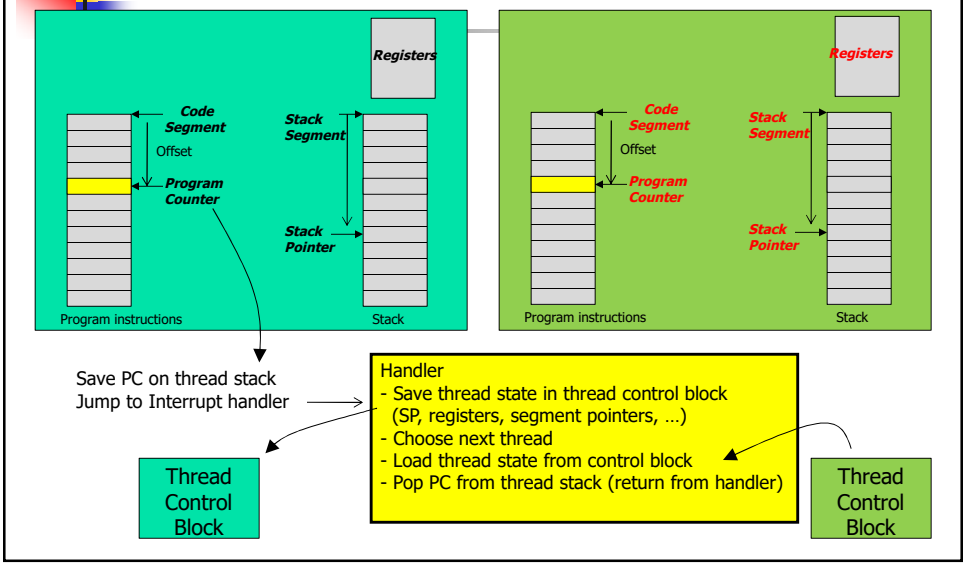
Context Switch (Interrupt)



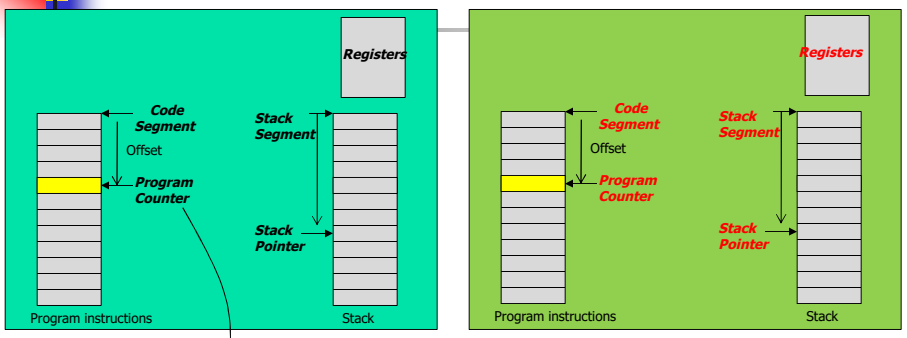
Context Switch (Interrupt)



Context Switch (Interrupt)



Context Switch (Interrupt)



Save PC on thread stack
Jump to Interrupt handler

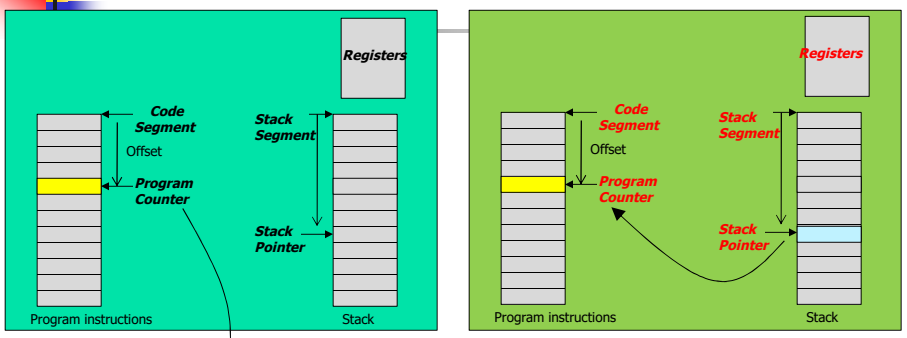
Handler
- Save thread state in thread control block (SP, registers, segment pointers, …)
- Choose next thread
- Load thread state from control block
- Pop PC from thread stack (return from handler)

Thread Control Block

Thread Control Block

Where does it return?

Context Switch (Interrupt)



Save PC on thread stack
Jump to Interrupt handler

Handler
- Save thread state in thread control block (SP, registers, segment pointers, …)
- Choose next thread
- Load thread state from control block
- Pop PC from thread stack (return from handler)

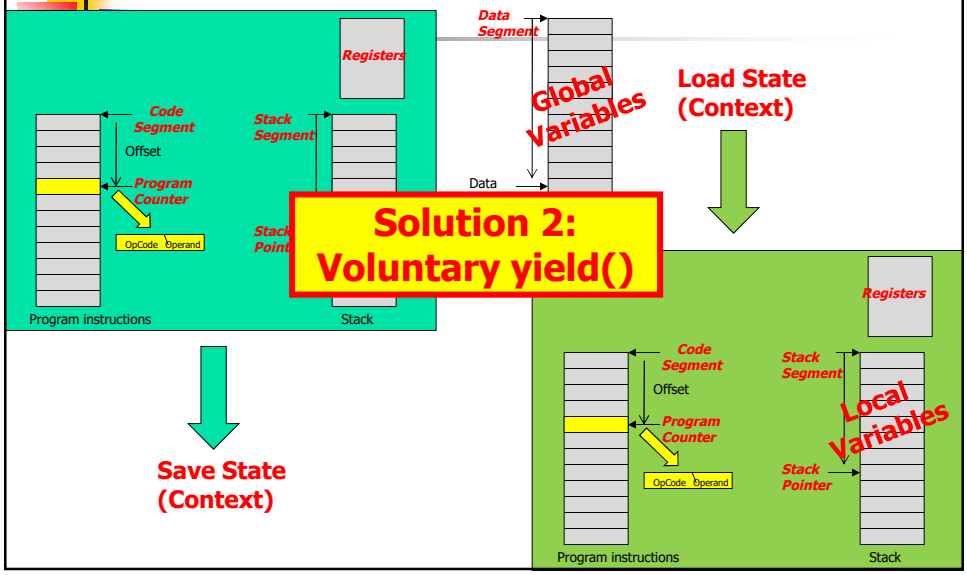
Thread Control Block

Thread Control Block

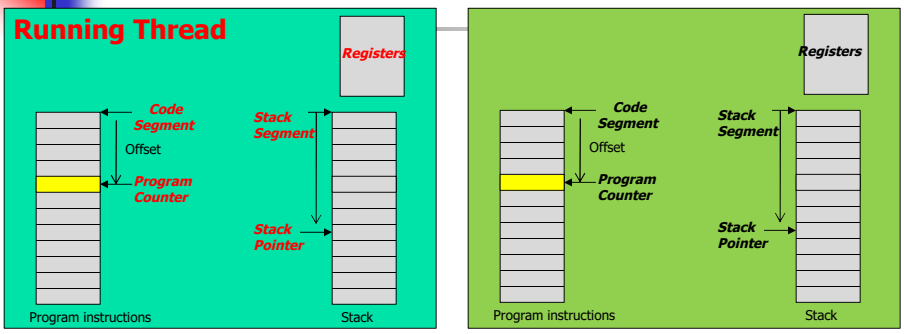
Where does it return?

Context Switch

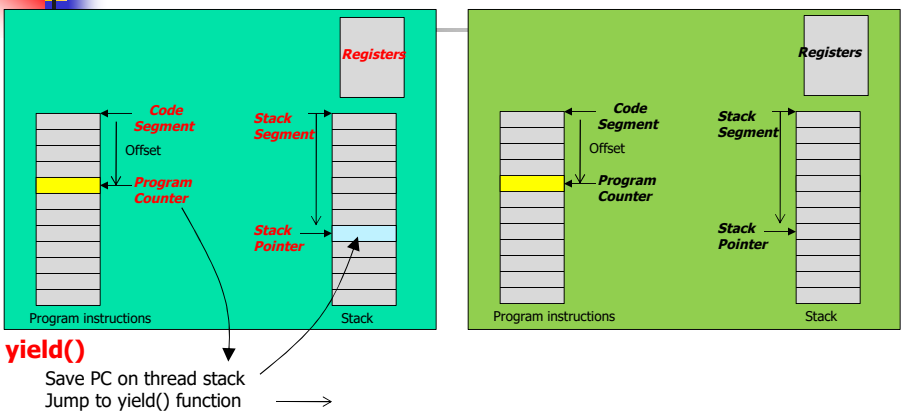
In thread context switches, heap is not switched



Context Switch (Yield)



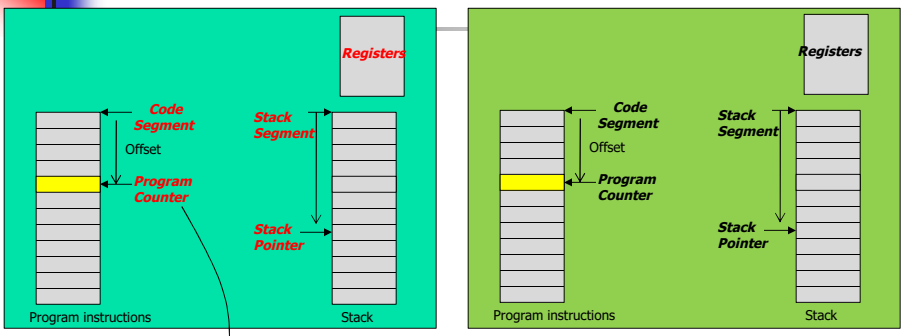
Context Switch (Yield)



yield()

Save PC on thread stack
Jump to yield() function

Context Switch (Yield)

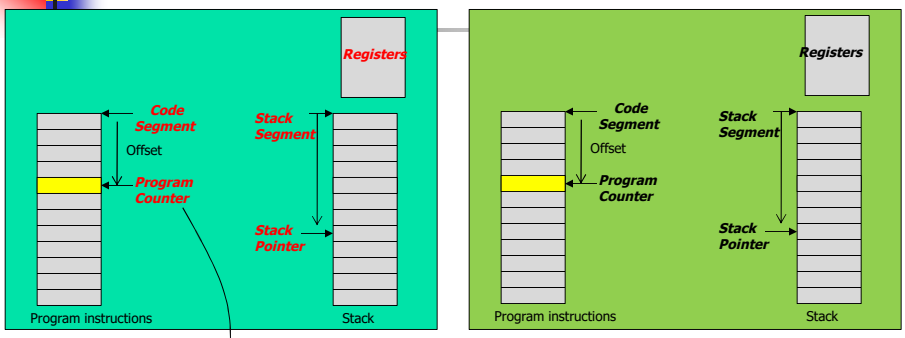


Save PC on thread stack
Jump to yield() function

yield()
- Save thread state in thread control block (SP, registers, segment pointers, …)

Thread Control Block

Context Switch (Yield)

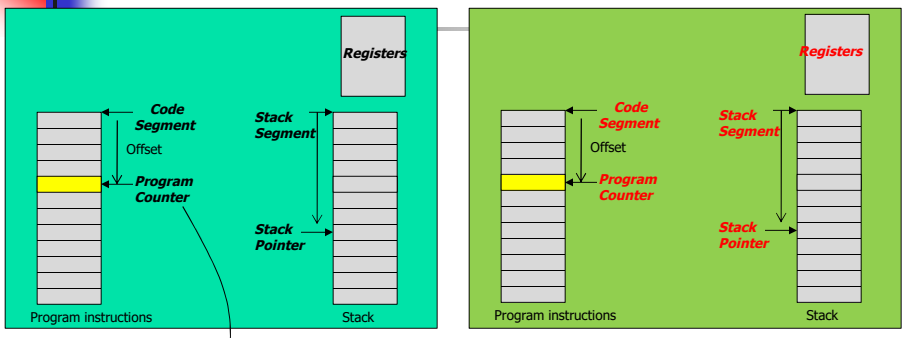


Save PC on thread stack
Jump to yield() function

yield()
- Save thread state in thread control block (SP, registers, segment pointers, …)
- Choose next thread

Thread Control Block

Context Switch (Yield)



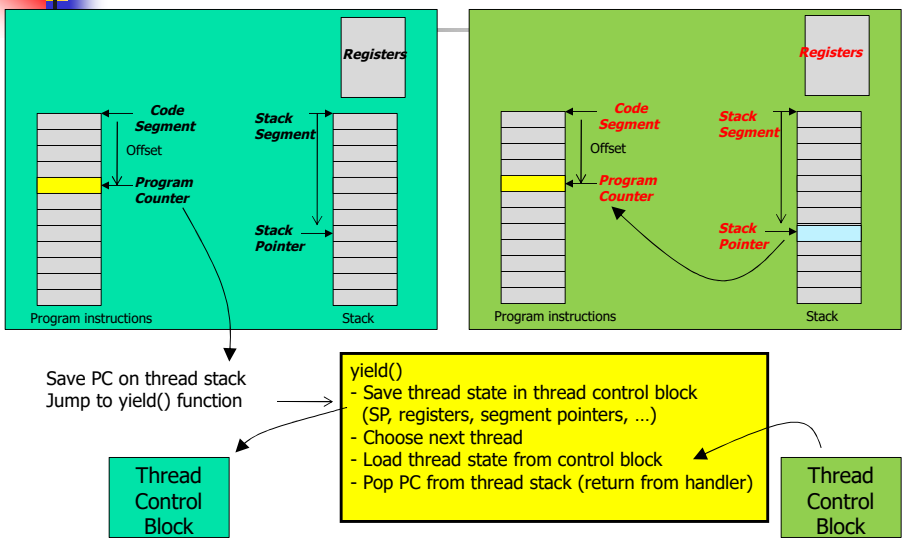
Save PC on thread stack
Jump to yield() function

yield()
- Save thread state in thread control block (SP, registers, segment pointers, …)
- Choose next thread
- Load thread state from control block

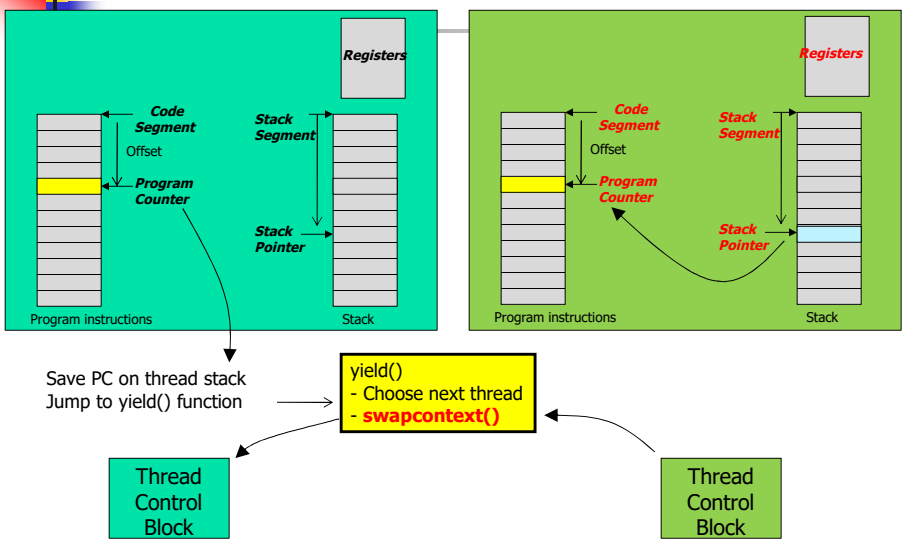
Thread Control Block

Thread Control Block

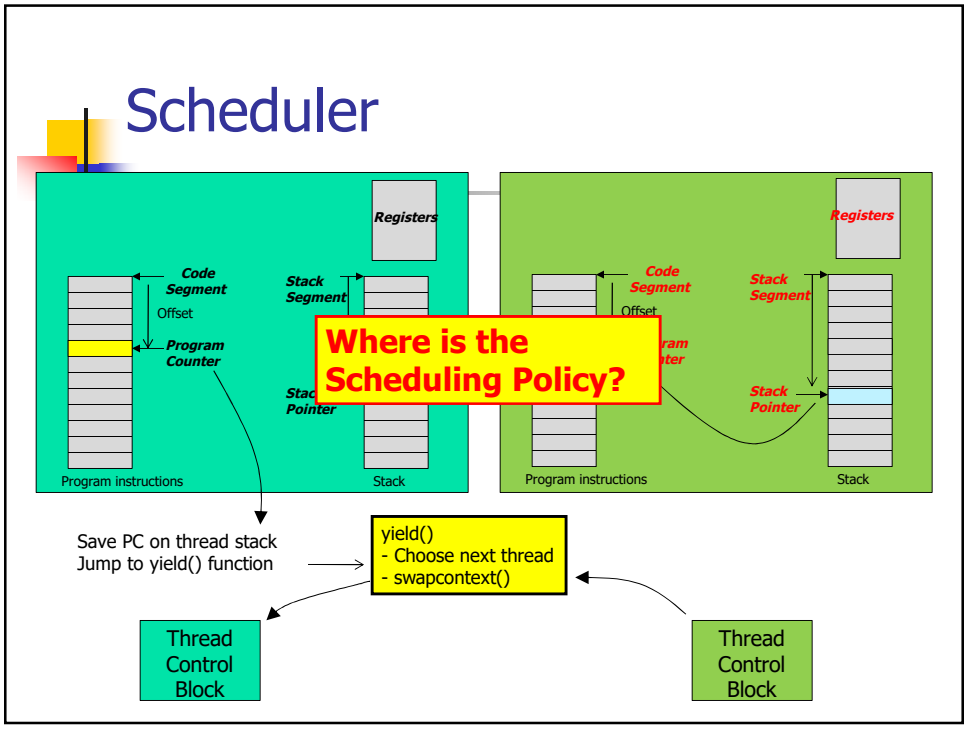
Context Switch (Yield)



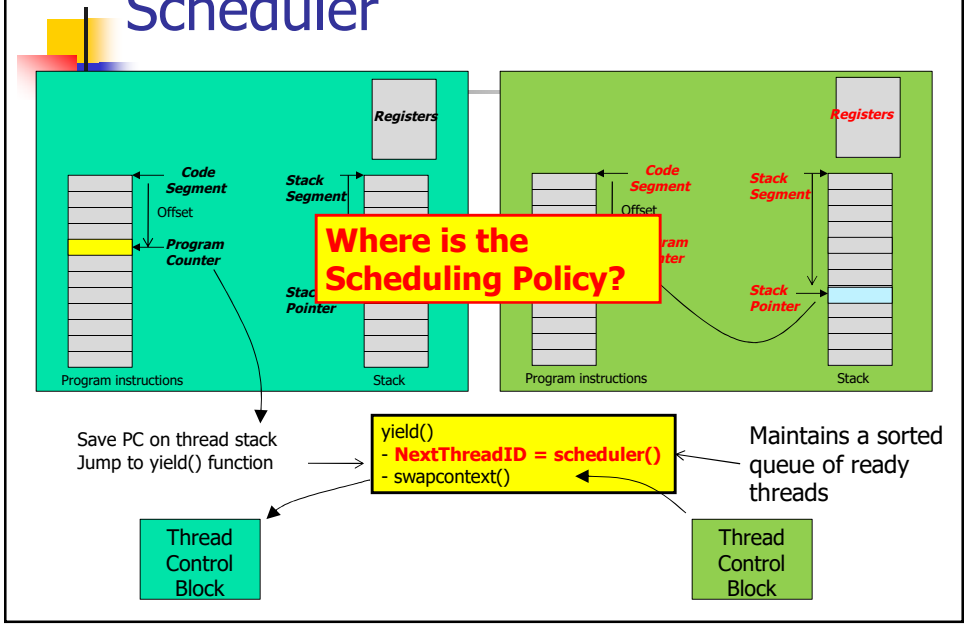
Linux User Level Implementation



Scheduler



Scheduler





Issues

- Initialization?
- Interrupt context switch versus yield?

33



Issues

- Initialization?
  - Stack for each thread needs to be allocated a priori → just malloc() the right sized chunk.
  - Thread control blocks must be created
- Interrupt context switch versus yield?

34



Issues

- Initialization?
  - Stack for each thread needs to be allocated a priori → just malloc() the right sized chunk.
  - Thread control blocks must be created
- Interrupt context switch versus yield?
  - Yield is voluntary. What if application did not yield?
  - Yield cannot do preemptive scheduling.

35

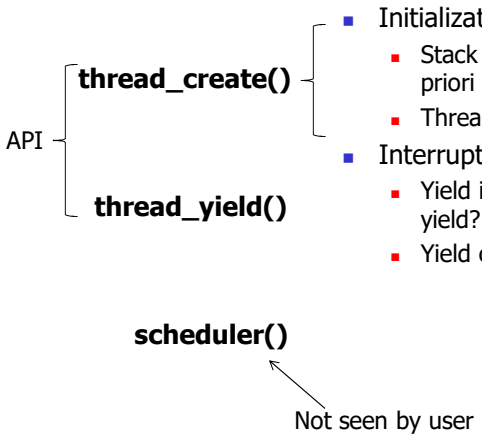


A User-level Thread Package

- Initialization?
  - Stack for each thread needs to be allocated a priori → just malloc() the right sized chunk.
  - Thread control blocks must be created
- Interrupt context switch versus yield?
  - Yield is voluntary. What if application did not yield?
  - Yield cannot do preemptive scheduling.

36

A User-level Thread Package



- Initialization?
  - Stack for each thread needs to be allocated a priori → just malloc() the right sized chunk.
  - Thread control blocks must be created
- Interrupt context switch versus yield?
  - Yield is voluntary. What if application did not yield?
  - Yield cannot do preemptive scheduling.

37

Review

- How do you implement a context switch?
  - In cooperative user-level threads?
  - In kernel threads?
- How do you implement a thread scheduler?
- Trade-offs between cooperative and non-cooperative thread models?

38