Project 2
Non-Preemptive Scheduling

COS 318

General Suggestions

- Use an IDE
  - Eclipse
    - Built into lab machines
      - Help -> Install New Software...
      - Download a specific Eclipse package for C/C++ from eclipse.org
  - Others
- Start as soon as you can and get as much done as possible by design review time
- bochsdbg / bochs-gdb

Good News

No more segments!

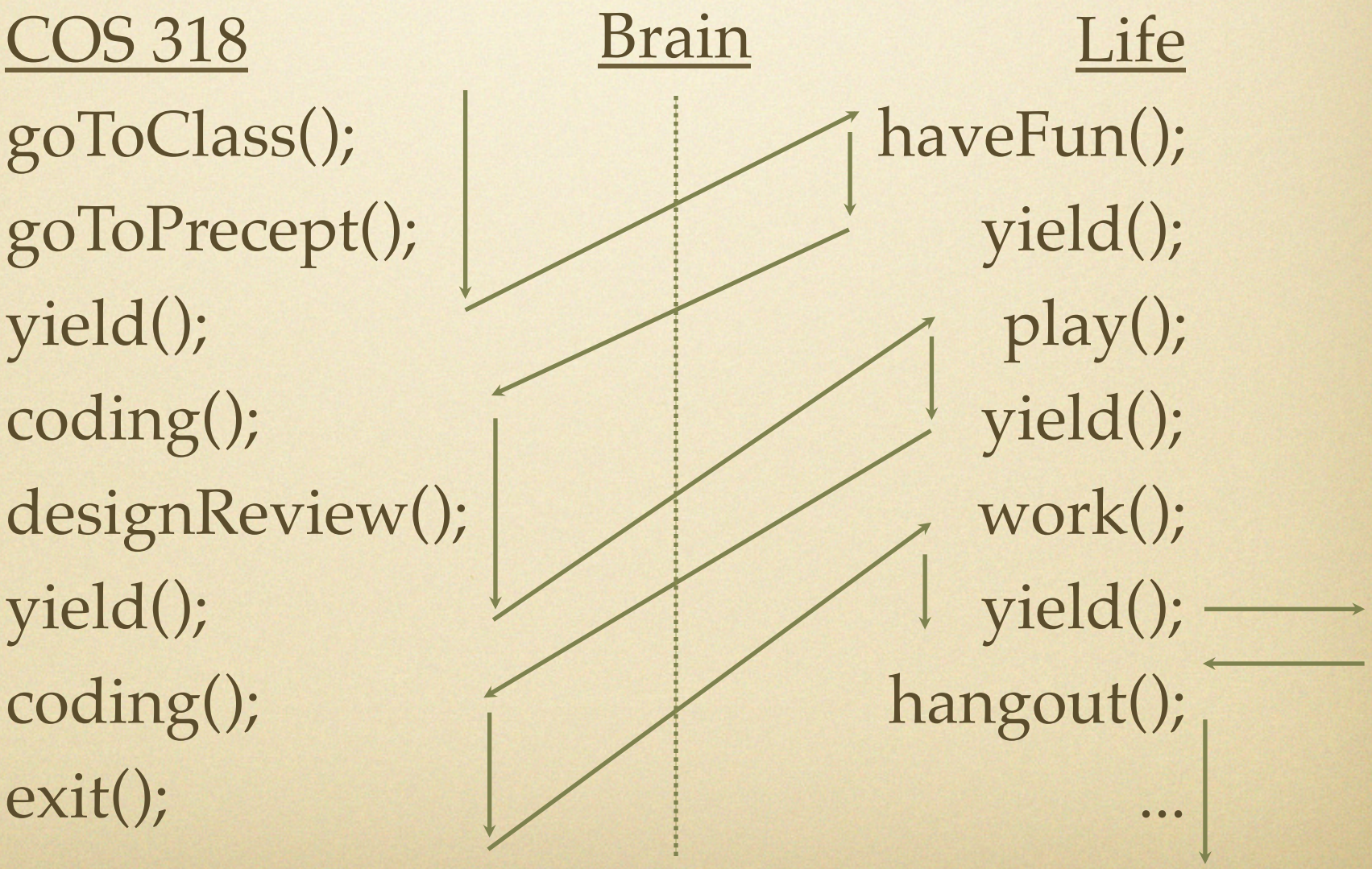
Overview

- Add multiprogramming to the kernel
  - Non-preemptive scheduler
    - 5 threads, 3 processes
      - Process Control Blocks
    - Context switching
      - Timing
- Mutual exclusion
  - Lock

Non-Preemptive

- What does it mean?
- yield & exit
  - do_yield() & do_exit() within the kernel (kernel threads can call these directly)
  - yield() & exit() for processes
    - dispatches a desire to call do_yield() or do_exit() to the kernel

Non-Preemptive Scheduling Example

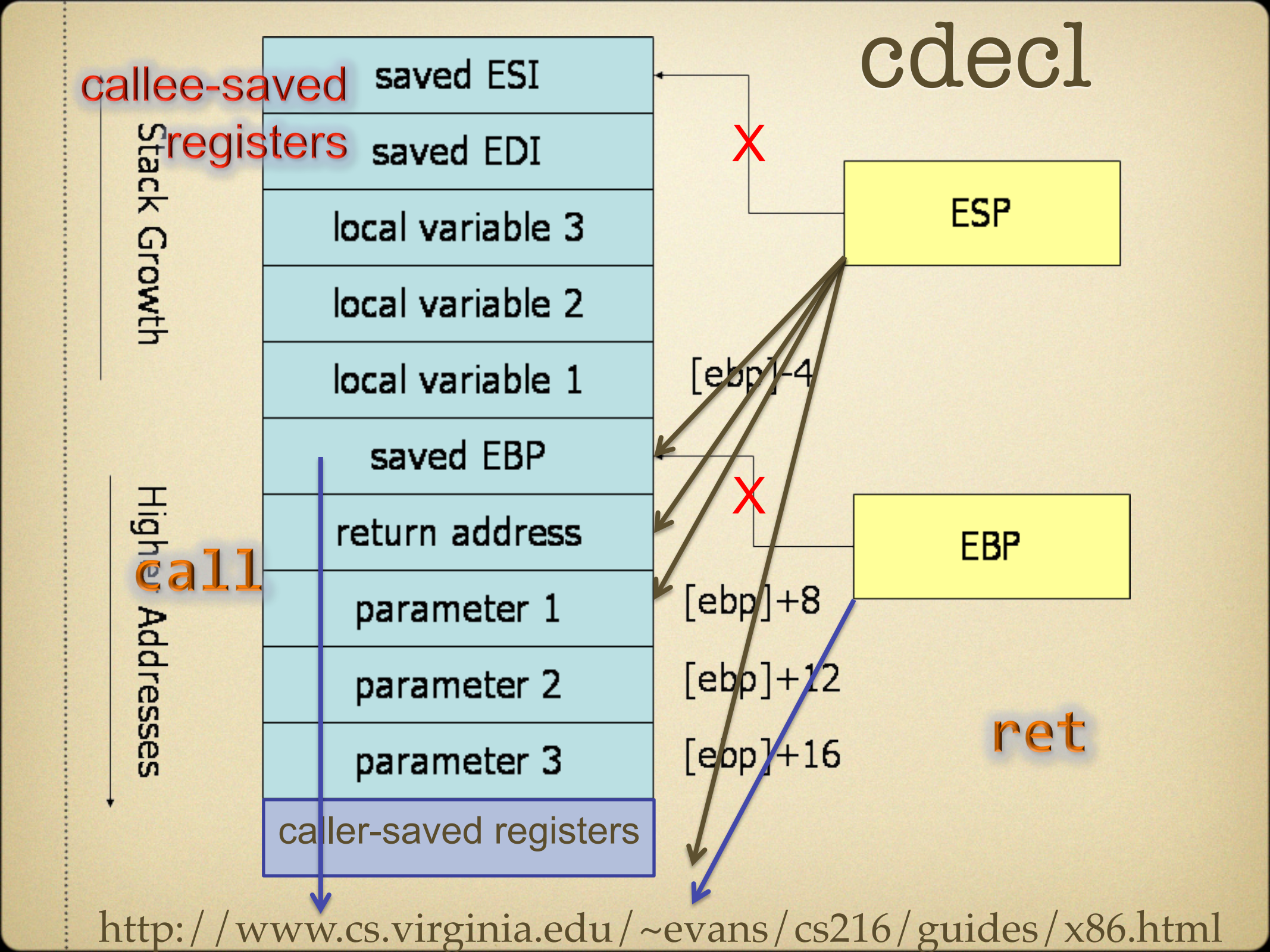


What yield'ing does

- When yield is called, the "context" of a task (thread or process) must be saved
- Process Control Block
  - What does it contain?
  - eflags (`pushfl`, `popfl`)
- Will be done in assembly
- Once the context is saved, the scheduler is run to pick a new task

Picking a New Task

- All tasks are waiting in queue
- Pick the next one from the front of the ready queue
- Restore it's state from the PCB
- `ret` to where the task was executing before


cdecl
callee-saved registers
saved ESI
saved EDI
local variable 3
local variable 2
local variable 1
saved EBP
return address
parameter 1
parameter 2
parameter 3
caller-saved registers
Stack Growth
Higher Addresses
call
X
ESP
[ebp]-4
X
EBP
[ebp]+8
[ebp]+12
[ebp]+16
ret
http://www.cs.virginia.edu/~evans/cs216/guides/x86.html

Why not just use `jmp`?

Mutual Exclusion

- Only one lock used by *threads*
- lock_init(lock_t * l)
- lock_acquire(lock_t * l)
- lock_release(lock_t * l)