Implementing UThreads: Data Structures

- **Thread State**
  - running: register file and runtime stack
  - stopped: Thread Control Block and runtime stack

- **Thread Control Block (TCB)**
  - thread status: (NASCENT, RUNNING, RUNNABLE, BLOCKED, or DEAD)
  - pointers to thread's stack base and top of stack
  - scheduling parameters such as priority, quantum, pre-emptibility etc.

- **Runnable Queue**
  - list of TCBs of all RUNNABLE threads
  - One or more Blocked Queues
  - list of TCBs of BLOCKED threads
Threads, Queues, and Execution Order

- Queue is confusing name!
- Scheduler may choose what to run next in any order
- Deterministic if uthread_init() called, nondeterministic with more processors

foo
bar
zot
join
bat

Priority, Round Robin Scheduling Policy

- Priority
  - A number assigned to each thread
  - Thread with highest priority goes first
- When choosing the next thread to run
  - Run highest priority runnable thread
  - When threads are same priority, run thread that has waited longest

Implementing Thread Mechanism

- Organize ready queue by priority
- Highest priority
- FIFO (first in first out) among threads of equal priority
- Benefits
  - Drawbacks and mitigation

Example Code for Thread Switch

```plaintext
Example Code for Thread Switch

Thread Yield
- Gets next runnable thread from ready queue (if any)
- Puts current thread on ready queue
- Switches to next thread
Example Code
```

Priority, Round Robin Scheduling Policy

- Priority
  - A number assigned to each thread
  - Thread with highest priority goes first
- When choosing the next thread to run
  - Run highest priority runnable thread
  - When threads are same priority, run thread that has waited longest

Implementing Thread Mechanism

- Organize ready queue by priority
- Highest priority
- FIFO (first in first out) among threads of equal priority
- Benefits
  - Drawbacks and mitigation
Summary

- **Thread**
  - synchronous "thread" of control in a program
  - virtual processor that can be stopped and started
  - threads are executed by real processor one at a time

- **Threads hide asynchrony**
  - by stopping to wait for interrupt/event, but freeing CPU to do other things

- **Thread state**
  - when running: stack and machine registers (register file etc.)
  - when stopped: Thread Control Block stores stack pointer, stack stores state

- **Round-robin, preemptive, priority thread scheduling**
  - lower priority thread preempted by higher
  - thread preempted when its quantum expires
  - equal-priority threads get fair share of processor, in round-robin fashion

Disassembly Strategy: Multiple Passes

- **1. find large-scale control flow**
  - draw arrows to notice patterns
    - if vs. while
    - if vs. if/else

- **2. find small-scale patterns: correspondences and symmetries across different spots in code**
  - variable usage: address given in .pos is read from and written to
  - push/pop function arguments on stack: inc/deca r5
  - function returns value: use of r0 value after function call
  - local variables within function offsets from r5

- **3. comment ASM line by line**

- **4. first pass from comments to verbose C**
  - don't worry about arrays vs variables vs structs: can't tell
  - assume all loops are while
  - avoid sign error: if/while (a) is opposite from beq

- **5. second pass to tighten**
  - can you eliminate temporary vars, e.g. inside loop?
  - can you turn verbose while loop into concise for?