# Administrative

- No class on Thursday
  - Start reading papers
- Proposals due on Saturday
- Not going to discuss virtual memory or address translation. Use the textbook if you aren’t familiar with these topics

- How to read a paper
- How to think about the discussions

## Second half class structure

- 15 minutes on quiz
- 15 minutes on background / presentation
- 15 minutes on discussion

# In class discussion

- Transition to reading / discussing papers
- Three default questions:
  - What did you like about the paper?
  - What did you dislike about the paper?
  - What future work did this inspire?
- Other questions designed to spark a discussion
  - Most papers are accepted / rejected based on opinions, rarely because of facts
  - This class will hopefully help you learn how your classmates think

# Thread switching in xv6

- When executing in kernel, executing on behalf of a thread
  - Kernel stack key to this abstraction on x86
    - Contains local state (stack) and process struct
    - E.g., current pointer

```
swtch(struct context *old,
      struct context *new)
```

# x86 assembly overview (32 bit)

- Movl src, dst
- Pushl reg
- Pushfl – push eflags
- Jump imm
- Popl reg
- Popfl – pop eflags

- Eax – gp reg
- Ebx – gp reg
- Ecx – gp reg
- Edx – gp reg
- …
- Ebp – frame pointer
- Eip -- PC
- Esp – stack pointer

# Thread switching in xv6

```
# void swtch(struct context *old, struct context *new);
#
# Save current register context in old
# and then load register context from new
swtch:
# Save old registers
movl 4(%esp), %eax # put old ptr into eax
popl 0(%eax) # save the old IP
movl %esp, 4(%eax) # and stack
movl %ebx, 8(%eax) # and other registers
movl %ecx, 12(%eax)
movl %edx, 16(%eax)
movl %esi, 20(%eax)
movl %edi, 24(%eax)
movl %ebp, 28(%eax)
```

## Thread switching in xv6

```
# Load new registers
movl 4(%esp), %eax # put new ptr into eax
movl 28(%eax), %ebp # restore other registers
movl 24(%eax), %edi
movl 20(%eax), %esi
movl 16(%eax), %edx
movl 12(%eax), %ecx
movl 8(%eax), %ebx
movl 4(%eax), %esp # stack is switched here
pushl 0(%eax) # return addr put in place
ret # finally return into new ctxt
```