[537] Threads

Tyler Harter
Review: Easy Piece 1

Virtualization

- CPU
- Memory
Review: Easy Piece 1

Virtualization

- CPU
  - Context Switch
  - Schedulers
- Memory
Review: Easy Piece 1

Virtualization

- CPU
  - Context Switch
  - Schedulers
- Memory
  - Allocation
  - Segmentation
  - Paging
Review: Easy Piece 1

Virtualization

CPU

- Context Switch
- Schedulers

Memory

- Allocation
- Segmentation
- Paging

- TLBs
- Multilevel
- Swapping
Review

(1) name 3 places xv6 stores registers for context switches?
(2) how can you game the multi-level feedback queue?
(3) what field does malloc need for a free node that a kernel doesn’t need to track a free page?
(4) which segment needs different translation rules? Why?
(5) how can a TLB avoid flushing for context switches
(6) what’s the advantage of multi-level PT over segmented PTs?
(7) what H/W support is needed to do LRU swapping?
Threads
(a) not interleaved

CPU: A A A B

Disk: A A

(b) interleaved

CPU: A B A B A B

Disk: A A
What if there is only one process?
Moor’s Law: # of transistors doubles every ~2 years
CPU Trends

The future:
- same speed
- more cores

Faster programs $\Rightarrow$ concurrent execution
Goal

Write applications that fully utilize many CPUs…
Strategy 1

Build applications from many communicating processes
- like Chrome (process per tab)
- communicate via pipe() or similar

Pros/cons?
Strategy 1

Build applications from many communicating processes
- like Chrome (process per tab)
- communicate via pipe() or similar

Pros/cons?
- don’t need new abstractions
- cumbersome programming
- copying overheads
- expensive context switching (why expensive?)
Strategy 2

New abstraction: the **thread**.

Threads are just **like processes**, but they share the **address space** (e.g., using same PT).
CPU 1
running thread 1

CPU 2
running thread 2

RAM
CPU 1
  running
  thread 1

CPU 2
  running
  thread 2

RAM
  PageDir A
  PageDir B
  ...

CPU 1
running thread 1
PTBR

CPU 2
running thread 2
PTBR

RAM
PageDir A
PageDir B
...
CPU 1
running thread 1

CPU 2
running thread 2

RAM
PageDir A
PageDir B
...

PTBR

IP

IP
Each thread may be executing different code at the same time.
CPU 1
running thread 1
PTBR
IP
SP

CPU 2
running thread 2
PTBR
IP
SP

RAM
PageDir A
PageDir B
...

Virt Mem (PageDir A)
CODE
HEAP
...

...
CPU 1
running thread 1
PTBR
IP
SP

CPU 2
running thread 2
PTBR
IP
SP

RAM
PageDir A
PageDir B
...

Virt Mem (PageDir A)
CODE
HEAP
STACK 1
STACK 2
threads executing different functions need different stacks
Demo: basic threads
Scheduling Problems (One CPU)

T1 0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

memory: 100
T1 eax: ?
T2 eax: ?
Scheduling Problems (One CPU)

```
0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4
```

memory:  100
T1 eax:  100
T2 eax:  ?
Scheduling Problems (One CPU)

T1 0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

T2

memory: 100
T1 eax: 100
T2 eax: 100
Scheduling Problems (One CPU)

T1
0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4

T2

memory:  100
T1 eax:  101
T2 eax:  100
Scheduling Problems (One CPU)

0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4

memory:  100
T1 eax:  101
T2 eax:  101
Scheduling Problems (One CPU)

0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4

memory: 101
T1 eax: 101
T2 eax: 101
Scheduling Problems (One CPU)

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

memory: 101
T1 eax: 101
T2 eax: 101
### Scheduling Problems (One CPU)

**State:**
- 0x9cd4: 100
- %eax: ?
- %rip = 0x195

**Thread 1**
- %eax: ?
- %rip: 0x195

**Thread 2**
- %eax: ?
- %rip: 0x195

**process control blocks:**

**T1**
- 0x195 mov 0x9cd4, %eax
- 0x19a add $0x1, %eax
- 0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

**State:**
- `0x9cd4`: 100
- `%eax`: 100
- `%rip` = 0x19a

**Thread 1**
- `%eax`: ?
- `%rip`: 0x195

**Thread 2**
- `%eax`: ?
- `%rip`: 0x195

**Process Control Blocks:**
- T1
  - `%eax` = 100
  - `%rip` = 0x19a

**Instructions:**
- T1
  - 0x195: `mov 0x9cd4, %eax`
  - 0x19a: `add $0x1, %eax`
  - 0x19d: `mov %eax, 0x9cd4`
Scheduling Problems (One CPU)

**Thread 1**
- `%eax`: ?
- `%rip`: 0x195

**Thread 2**
- `%eax`: ?
- `%rip`: 0x195

**State**
- `0x9cd4`: 100
- `%eax`: 101
- `%rip` = 0x19d

**Process Control Blocks**
- `%eax`: ?
- `%rip`: 0x195

**Instruction Sequence**
- `0x195`: `mov 0x9cd4, %eax`
- `0x19a`: `add $0x1, %eax`
- `0x19d`: `mov %eax, 0x9cd4`
Scheduling Problems (One CPU)

State:
0x9cd4: 101
%eax: 101
%rip = 0x1a2

Thread 1
%eax: ?
%rip: 0x195

Thread 2
%eax: ?
%rip: 0x195

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

T1
Thread 1
%eax: ?
%rip: 0x195

Thread 2
%eax: ?
%rip: 0x195

State:
0x9cd4: 101
%eax: 101
%rip = 0x1a2

Context Switch

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

T1 →
Scheduling Problems (One CPU)

State:
0x9cd4: 101
%eax: ?
%rip = 0x195

Thread 1
%eax: 101
%rip: 0x1a2

Thread 2
%eax: ?
%rip: 0x195

Context Switch

T2
0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

**State:**
- $0x9cd4$: 101
- `%eax`: ?
- `%rip = 0x195`

**T2**
- $0x195$
- $0x19a$
- $0x19d$

**Thread 1**
- `%eax: 101`
- `%rip: 0x1a2`

**Thread 2**
- `%eax: ?`
- `%rip: 0x195`

```
mov 0x9cd4, %eax
add $0x1, %eax
mov %eax, 0x9cd4
```
Scheduling Problems (One CPU)

State:
0x9cd4: 101
%eax: 101
%rip = 0x19a

Thread 1
%eax: 101
%rip: 0x1a2

Thread 2
%eax: ?
%rip: 0x195

process control blocks:

T2
0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

**State:**
- $0x9cd4$: 101
- %eax: 102
- %rip = 0x19d

Thread 1
- %eax: 101
- %rip: 0x1a2

Thread 2
- %eax: ?
- %rip: 0x195

**Process Control Blocks:**
- T2
  - %eax: 101
  - %rip: 0x1a2

**Instructions:**
- 0x195 mov $0x9cd4, %eax
- 0x19a add $0x1, %eax
- 0x19d mov %eax, 0x9cd4

T2
Scheduling Problems (One CPU)

State:
0x9cd4: 102
%eax: 102
%rip = 0x1a2

Thread 1:
%eax: 101
%rip: 0x1a2

Thread 2:
%eax: ?
%rip: 0x195

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

T2
Scheduling Problems (One CPU)

State:
- \texttt{0x9cd4}: 102
- \%eax: 102
- \%rip = \texttt{0x1a2}

GOOD!

Thread 1:
- \%eax: 101
- \%rip: \texttt{0x1a2}

Thread 2:
- \%eax: ?
- \%rip: \texttt{0x195}

\texttt{0x195} \quad \text{mov} \quad \texttt{0x9cd4}, \quad \%eax

\texttt{0x19a} \quad \text{add} \quad \$0x1, \quad \%eax

\texttt{0x19d} \quad \text{mov} \quad \%eax, \quad \texttt{0x9cd4}

T2
Another schedule
Scheduling Problems (One CPU)

**State:**
- `0x9cd4`: 100
- `%eax`: ?
- `%rip` = `0x195`

Thread 1
- `%eax`: ?
- `%rip`: `0x195`

Thread 2
- `%eax`: ?
- `%rip`: `0x195`

process control blocks:

T1
- `0x195` mov `0x9cd4`, `%eax`
- `0x19a` add `$0x1`, `%eax`
- `0x19d` mov `%eax`, `0x9cd4`
Thread 1
\%eax: ?
\%rip: 0x195

Thread 2
\%eax: ?
\%rip: 0x195

State:
\texttt{0x9cd4}: 100
\texttt{\%eax}: 100
\texttt{\%rip} = \texttt{0x19a}

\textbf{T1} \quad \begin{array}{ll}
\texttt{0x195} & \texttt{mov} \ \texttt{0x9cd4}, \ \texttt{\%eax} \\
\texttt{0x19a} & \texttt{add} \ \texttt{$0x1, \ \%eax} \\
\texttt{0x19d} & \texttt{mov} \ \texttt{\%eax, 0x9cd4}
\end{array}
Scheduling Problems (One CPU)

**State:**
- 0x9cd4: 100
- %eax: 101
- %rip = 0x19d

**Thread 1**
- %eax: ?
- %rip: 0x195

**Thread 2**
- %eax: ?
- %rip: 0x195

**Process Control Blocks:**
- T1

**Assembly Code:**
- 0x195 mov 0x9cd4, %eax
- 0x19a add $0x1, %eax
- 0x19d mov %eax, 0x9cd4
### Scheduling Problems (One CPU)

#### State:
- `0x9cd4`: 100
- `%eax`: 101
- `%rip` = 0x19d

#### Context Switch
- `0x195` mov `0x9cd4, %eax`
- `0x19a` add `$0x1, %eax`
- `0x19d` mov `%eax, 0x9cd4`

#### Thread 1
- `%eax`: ?
- `%rip`: 0x195

#### Thread 2
- `%eax`: ?
- `%rip`: 0x195

---

**Diagram:**

- **Thread 1**
  - `%eax`: 101
  - `%rip`: 0x19d

- **Thread 2**
  - `%eax`: ?
  - `%rip`: 0x195

---

**Process Control Blocks:**

- **T1**
  - `%eax`: ?
  - `%rip`: 0x195

---

**Code Snippet:**

```
0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4
```
Scheduling Problems (One CPU)

State:
0x9cd4: 100
%eax: ?
%rip = 0x195

Context Switch

T2
0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

**State:**
- 0x9cd4: 100
- %eax: ?
- %rip = 0x195

Thread 1:
- %eax: 101
- %rip: 0x19d

Thread 2:
- %eax: ?
- %rip: 0x195

Thread 2

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

process control blocks:
Scheduling Problems (One CPU)

State:
- $0x9cd4$: 100
- $%eax$: 100
- $%rip = 0x19a$

T2
- $0x195$
- $0x19a$
- $0x19d$

Code:
- $0x195$: `mov $0x9cd4, %eax`
- $0x19a$: `add $0x1, %eax`
- $0x19d$: `mov %eax, $0x9cd4`

Thread 1
- $%eax$: 101
- $%rip$: 0x19d

Thread 2
- $%eax$: ?
- $%rip$: 0x195
Scheduling Problems (One CPU)

**State:**
- \(0x9cd4\): 100
- \(%eax\): 101
- \(%rip\) = 0x19d

**Thread 1**
- \(%eax\): 101
- \(%rip\): 0x19d

**Thread 2**
- \(%eax\): ?
- \(%rip\): 0x195

---

**Thread 2** (T2)
- 0x195: mov \(0x9cd4\), \(%eax\)
- 0x19a: add $0x1, \(%eax\)
- 0x19d: mov \(%eax\), \(0x9cd4\)
Scheduling Problems (One CPU)

**State:**
- $0x9cd4$: 101
- $%eax$: 101
- $%rip = 0x1a2$

**Thread 1**
- $%eax: 101$
- $%rip: 0x19d$

**Thread 2**
- $%eax: ?$
- $%rip: 0x195$

---

**T2**

- $0x195$: `mov 0x9cd4, %eax`
- $0x19a$: `add $0x1, %eax`
- $0x19d$: `mov %eax, 0x9cd4`
Scheduling Problems (One CPU)

**State:**
- 0x9cd4: 101
- %eax: 101
- %rip = 0x1a2

**Thread 1**
- %eax: 101
- %rip: 0x19d

**Thread 2**
- %eax: ?
- %rip: 0x195

**Context Switch**

```
0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4
```

T2
Scheduling Problems (One CPU)

State:
0x9cd4: 101
%eax: 101
%rip = 0x19d

Context Switch
0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

Thread 1
%eax: 101
%rip: 0x19d

Thread 2
%eax: 101
%rip: 0x1a2
Scheduling Problems (One CPU)

**State:**
- \(0x9cd4: 101\)
- \(\%eax: 101\)
- \(\%rip = 0x19d\)

**T1**
- \(0x195\) mov \(0x9cd4, \%eax\)
- \(0x19a\) add \(0x1, \%eax\)
- \(0x19d\) mov \(\%eax, 0x9cd4\)

**Thread 1**
- \(\%eax: 101\)
- \(\%rip: 0x19d\)

**Thread 2**
- \(\%eax: 101\)
- \(\%rip: 0x1a2\)
Scheduling Problems (One CPU)

State:
0x9cd4: 101
%eax: 101
%rip = 0x1a2

Thread 1
%eax: 101
%rip: 0x19d

Thread 2
%eax: 101
%rip: 0x1a2

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4
### Scheduling Problems (One CPU)

#### State:
- `$0x9cd4`: 101
- `%eax`: 101
- `%rip = 0x1a2`

#### Thread 1:
- `%eax: 101`
- `%rip: 0x19d`

#### Thread 2:
- `%eax: 101`
- `%rip: 0x1a2`

#### Wrong!

- **T1**
  - `0x195` mov `$0x9cd4, %eax`
  - `0x19a` add `$0x1, %eax`
  - `0x19d` mov `%eax, $0x9cd4`
Thread 1
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

Thread 2
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

How much is added?
Timeline View

Thread 1
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

Thread 2
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

How much is added?
Timeline View

**Thread 1**
- mov 0x123, %eax
- add %0x1, %eax
- mov %eax, 0x123

**Thread 2**
- mov 0x123, %eax
- add %0x1, %eax
- mov %eax, 0x123

How much is added?
Thread 1
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

Thread 2
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

How much is added?
Thread 1

mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

Thread 2

mov 0x123, %eax
add %0x1, %eax

mov %eax, 0x123

How much is added?
Exercise

Thread 1
(a) mov 0x123, %eax
(b) add %0x3, %eax
(c) mov %eax, 0x123

Thread 2
(d) mov 0x123, %eax
(e) mult %0x2, %eax
(f) mov %eax, 0x123

Say int at 0x123 is 10. Find interleavings to produce the following: 13, 20, 23, 26
Concurrency leads to non-deterministic bugs, called race conditions.

Whether bug manifests depends on CPU schedule!

Passing tests means little.

How to program: imagine scheduler is malicious.
What do we want?

Want all or none of these instructions to execute. That is, we want them to be atomic.

`mov 0x123, %eax`
`add %0x1, %eax`
`mov %eax, 0x123`
What do we want?

Want all or none of these instructions to execute. That is, we want them to be atomic.

```
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123
```
What do we want?

Want all or none of these instructions to execute. That is, we want them to be atomic.

```
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123
```

critical section

We want mutual exclusion for critical sections. That is, if I run, you can’t (and vice versa).