Process A's Kernel Stack:
- %esp
- %eip (all traps)
- trap locals
- %eip (trap)
- yield locals
- %eip (yield)
- sched locals
- cpu -> scheduler
- &proc -> context
- %eip (sched)
- %ebp
- %ebx
- %esi
- %edi

Scheduler Stack:
- &proc -> context
- cpu -> scheduler
- %eip (scheduler)
- %ebp
- %ebx
- %esi
- %edi

%eax

%edx

%esp

NOTE: Arrows (→) indicate the state after switching from A to scheduler.
Context Switch
From scheduler to Proc B

Scheduler stack

Process B's kernel stack

---

NOTE: Arrow indicate the state after switching from scheduler to B.
Review

- Process - PCB - process table
  - Limited Direct Execution.
    - System calls
    - Timer interrupts.
- Context Switch.
- Scheduling policies (MLFQ).

```
shell

> ls
  a.c  b.c
>
```

```
/bin
/usr/bin

parent

fork()

child

shell

pid = 2

shell

pid = 3
```
Apps.

fork() → exit() → read() → write()

OS

parent

data

code

PC 1

child

data

code

PC 1
fork()

execvp("ls", args)

same child
Virtualizing

16KB

A

stack

↓

FREE

↑

heap

data

2KB

code

0KB

Process' address space

1. contiguous in phy. mem
2. addr space < phy mem
3. all process have 16 KB of addr space.

64 KB

48 KB

32 KB

16 KB

OS

Instr @ addr 128

\[ 16\text{KB} = 2 \times 2^{10} = 2^{14} \]

\[ 2^{32} = 4 \text{ GB} \]
1. Virtualization

2. Concurrency

3. Persistence

\[ \text{Virt addr} = 128 \]
\[ \text{phy. addr} = 32KB + 128 \]
\[ = 32 \times 1024 \]
void fn ( ) {
    int X = 100;
    X = X + 3;
}

128: movl (%ebx), %eax
132: addl $0x03, %eax
135: movl %eax, (%ebx)

OX1000

100

X

%eax

pC

132

1. fetch instr. @ addr 128.
2. fetch data from addr (%ebx).
virt addr: 128

phy addr = virt.addr + base

= 128 + 32KB (32768) 

phy addr = 32896
Dynamic / HW-based / Base-Bounds Relocation.

CPU

MMU

32 KB
Base

16 KB
Bounds

phy. addr = virt. addr + base

128: movl (%ebx), %eax

phy. addr = \(128 + 32\text{KB}\)

\[32768\]

phy. addr = 32796

Load x from 15 KB

\[\frac{1024}{32} = 32\]

\[\frac{2048}{32} = 64\]

\[\frac{3072}{32} = 96\]

\[\frac{32768}{32} = 1024\]

\[x\text{'s phy. addr} = 15\text{KB} + 32\text{KB}\]

\[= 47\text{KB}\]

Bounds / Limit Register?

Memory protection.

"exceptions"

Dynamic Relocation - HW requirements:

1. Privileged mode
2. Base/bounds registers
3. Address translation & bounds check
4. Ability to raise exceptions
5. Privileged instr < update base/bounds
6. Privileged < register exception handler.
OS concerns:
1. Free list (16KB - 32KB, 48-64KB) → allocate memory for new process from this list.
2. Process terminates — free phy. memory & add it to free list
3. Context switching
   - save base/bounds pair in PCB
   - restore pointer from n.
   * OS can stop the process & move its addr. space to a diff. spot in phy. mem.

LDE (Dynamic Relocation) — Slides.

Problems?
- internal fragmentation.
<table>
<thead>
<tr>
<th>Seg</th>
<th>Base</th>
<th>Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>32 KB</td>
<td>2 KB</td>
</tr>
<tr>
<td>heap</td>
<td>34 KB</td>
<td>2 KB</td>
</tr>
<tr>
<td>slack</td>
<td>28 KB</td>
<td>2 KB</td>
</tr>
</tbody>
</table>

1. virt addr 100 in code segment

   phy. addr = 32 KB + 100
   = 32868

2. virt addr 4200 in heap segment

   phy. addr = 34 KB + (4200 - 4096)
   = 104
   = 34920

3. virt addr 7 KB

   Illegal mem access - "Seg Fault"
16 KB = $2^4 \times 2^{10}$ bytes

= 2^14 bytes

\[ \text{14 bits} \]

\[ \text{virt. addr} = 15 \text{ KB} \]

\[ \text{phy. addr} = 28 \text{ KB} + (3 \text{ KB} - 4 \text{ KB}) \]

\[ = 28 \text{ KB} - 1 \text{ KB} \]

\[ = 27 \text{ KB} \]
Which Segment are we referring to?

13 12 11 10 9 8 7 6 5 4 3 2 1 0

Segment: offset

16 | 4200
   | 262 - 8
   | 16 - 6
   | - 0

00 - code
01 - heap
101 - stack.

\[
\begin{align*}
1024 & \div 2^4 \\
34 & \\
4096 & \div 2^6 \\
3072 & + 104 = 34920
\end{align*}
\]

0x068 = 8 + 16x6 = 104

2^12 = 4KB = 2^2 \times 2^{10}

48 + 8 = 56