1. HARDWARE/OS BASICS

ASSUMPTIONS
- Every device is accessed through the memory (Memory mapped devices)
- There are special areas in memory for access to devices

REQUIREMENTS
- To support multiple users/processors,
  - Need to restrict access to – RAM, Registers
  - Keep one process from running for ever
- Segment registers to mediate access to RAM called the TLB (Translation Look-aside Buffer)

TLB
- Maps Virtual memory address to physical location in RAM
- Many-to-one mapping, out of order addressing possible
- The memory is divided in units of pages. Assuming 4Kb pages, a location in the page can be addressed with 12bits.
- If the page is not found, a page fault occurs

Fig. Simple machine hardware
Virtual (20bit) Physical (20bit)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
</tr>
</tbody>
</table>

Fig. TLB table

2. Model for Executing Two Processes with TLB

Proposed Model

- At Boot Time OS has access to all of memory
- Multiple modes of operation (As in x86 systems): OS starts in real mode addressing memory using physical address
- Add entries for itself to TLB (Note: OS needed in TLB for error handling code, page faults etc.)
- Switch into protected addressing mode and load App A, App B TLB entries.

Issues

- App A can access OS, A, and B memory areas and vice versa
- If only App A in TLB, it can add entries to TLB.

Solution

- Only OS should be privileged to manipulate TLB
- CPU needs to know if it is running privileged code
- Privilege bit in CPU [PRIV]. Controls whether TLB instruction/calls to interrupt handler are allowed
- Potential PRIV bit transition problem (transitioning from PRIV=0 → PRIV=1). Application should not be allowed to transition to OS mode. Solution: Use clocked interrupts
- To run App A
  1) Clear old entries from TLB
  2) Add App A entries to TLB
  3) Set Clock interrupt 100ms, jump to IH and set priv=0
  4) Jump to app A
  5) Interrupt after 100ms restores PRIV =1
- To reload OS stub entries are kept in the TLB (OS Stub entries have no secret data)
- Introduce Read/Write/Execute permission bits in the TLB to prevent data corruption by Apps. Only OS is allowed to modify the read write bits.
- Security goals achieved:
  a. Integrity: Through TLB (Read only for Apps)
  b. Availability: Ensured through the Clock which interrupts the application execution to ensure availability to other apps.
  c. Confidentiality: OS clears the existing App information from the TLB before loading TLB entries for a new App. Also, the OS stub entries in TLB do not have any secret data.
ALTERNATE DESIGN FOR TLB

- Replace PRIV bit by Process IDs (PID) to maintain process id state of the currently running process in the CPU with Read/Write/Execute bits for specific process debugging purposes.

<table>
<thead>
<tr>
<th>Pid</th>
<th>Ops</th>
<th>Virtual</th>
<th>Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (OS)</td>
<td>X</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>RWX</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>RWX</td>
<td>12</td>
<td>2</td>
</tr>
</tbody>
</table>

*Fig. TLB with PID*

- No need to clear TLB frequently, app entries can exist together
- Application calls system call which in turn generates an interrupt
- Need specific interrupt handlers