Optimizing address translation (TLB)

- Avoid two memory accesses for each read or write
- Rely on locality of reference to the page table
- Translation Lookaside Buffer (TLB) - cache holding page table mappings
- TLB tag: higher bites of the virtual address
- Valid bit, reference bit, dirty bit

- TLB size: 32-4096 entries
- Block size: 1-2 page table entries
- Hit time: 0.5-1 clock cycle
- Miss penalty: 10-30 clock cycles
- Miss rate: 0.01% - 1%
- Associativity: various (e.g. random)
DECStation 3100: TLB and cache organization

Virtual address

31 30 29  .........  15 14 13 12 11 10 9 8  3 2 1 0

Virtual page number  Page offset

Valid Dirty Tag  Physical page number

TLB hit

TLB

Physical page number

Physical page number  Physical address tag  Page offset

Physical address tag  Cache index  Byte offset

Cache

Valid  Tag  Data

Cache hit

Data

Byte offset
DECStation 3100: processing read or write

Virtual address

TLB access

TLB hit?

No

TLB miss exception

Yes

Physical address

Write?

No

Try to read data from cache

Cache miss stall

Cache hit?

No

Deliver data to the CPU

Yes

Write data into cache, update the tag, and put the data and the address into the write buffer

Write access bit on?

No

Write protection exception

Yes

Cache hit?

No

TLB access

Yes

Deliver data to the CPU
## Overall operation of a memory hierarchy

<table>
<thead>
<tr>
<th>Cache</th>
<th>TLB</th>
<th>Page table</th>
<th>Possible? If so under what circumstance</th>
</tr>
</thead>
<tbody>
<tr>
<td>miss</td>
<td>hit</td>
<td>hit</td>
<td>Possible, but never detected (page table is not checked after a TLB hit).</td>
</tr>
<tr>
<td>hit</td>
<td>miss</td>
<td>hit</td>
<td>TLB misses, but entry found in page table. After retry, found in cache.</td>
</tr>
<tr>
<td>miss</td>
<td>miss</td>
<td>hit</td>
<td>TLB misses, but entry found in page table. After retry, miss in cache.</td>
</tr>
<tr>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>TLB misses and is followed by page fault. After retry, miss in cache.</td>
</tr>
<tr>
<td>miss</td>
<td>hit</td>
<td>miss</td>
<td>Impossible: cannot have translation in TLB if page is not in memory.</td>
</tr>
<tr>
<td>hit</td>
<td>hit</td>
<td>miss</td>
<td>Impossible: cannot have translation in TLB if page is not in memory.</td>
</tr>
<tr>
<td>hit</td>
<td>miss</td>
<td>miss</td>
<td>Impossible: data cannot be in cache if page is not present in memory.</td>
</tr>
</tbody>
</table>
Handling TLB misses and page faults

**TLB miss** (handled in software or hardware):
- The page is in memory, but its physical address is missing. A new TLB entry must be created
- The page is not in memory and the control is transferred to the operating system to deal with a page fault

**Page fault** (handled by the operating system):
- Causing exception (interrupt): using EPC and Cause register
- Instruction page fault:
  - Store the state of the process
  - Look up the page table to find the disk address of the referenced page
  - Choose a physical page to replace
  - Start a read from disk for the referenced page
  - Execute another process until the read completes
  - Restart the instruction which caused the fault
- Data access page fault:
  - Occurs in the middle of an instruction
  - MIPS instructions are restartable: prevent the instruction from completing and restart it from the beginning
  - More complex machines: interrupting instructions (saving the state of CPU)