Caching vs. Demand Paging

**Caching**
- cache slot (~32 bytes)
- cache line (~32 bytes)
- cache miss rate (1% to 20%)
- cache hit (~1 cycle)
- cache miss (~100 cycles)
- miss is handled in *hardware*

**Demand paging**
- page frame
- page (~4K bytes)
- page miss rate (<0.001%)
- page hit (~100 cycles)
- page miss (~5M cycles)
- miss is handled mostly in *software*
Address Translation: putting it all together

Virtual Address → TLB Lookup

- TLB miss: Page Table Walk
- TLB hit: Protection Check

- Protection Check: 
  - Protection denied → Physical Address (to cache)
  - Protection permitted: the page is in memory

Page Fault (OS loads page) → Update TLB

Address Translation in Pipeline Machines

- PC → Inst TLB → Inst. Cache → Decode → M → Data TLB → Data Cache → W

- TLB miss? Page Fault? Protection violation?

- Software handlers need a restartable exception on page fault or protection violation
- Handling a TLB miss needs a hardware or software mechanism to refill TLB
- Methods to overcome the additional latency of a TLB:
  - slow down the clock
  - pipeline the TLB and cache access
  - virtual address caches
  - parallel TLB/cache access
Physical vs Virtual Address Caches?

One cycle in case of a hit (+)
- cache needs to be flushed on a context switch unless address space identifiers (ASIDs) included in tags (-)
- aliasing problems due to the sharing of pages (-)

Alternative: place the cache before the TLB

Aliasing in Virtual-Address Caches

Two virtual pages share one physical page

Virtual cache can have two copies of same physical data.
Writes to one copy not visible to reads of other!

General Solution: Disallow aliases to coexist in cache

Software (i.e., OS) solution for direct-mapped cache

VAs of shared pages must agree in cache index bits; this ensures all VAs accessing same PA will conflict in direct-mapped cache (early SPARC)
Concurrent Access to TLB & Cache

Index L is available without consulting the TLB
⇒ cache and TLB accesses can begin simultaneously
Tag comparison is made after both accesses are completed

Cases:
- \( L + b = k \)
- \( L + b < k \)
- \( L + b > k \) what happens here? Partially VA cache!

Virtual-Index Physical-Tag Caches: Associative Organization

After the PPN is known, W physical tags are compared
Allows cache size to be greater than \( 2^{L+b} \) bytes
Exception handling in a pipeline machine

1. An instruction may cause multiple exceptions; which one should we process? from the earliest stage
2. When multiple instructions are causing exceptions; which one should we process first? from the oldest instruction
### Interrupt processing

- Internal interrupts can happen at any stage but cause a redirection only at Commit.
- External interrupts are considered only at Commit.
- If an instruction causes an interrupt then the external interrupt, if present, is given a priority and the instruction is executed again.
  - Some instructions, like Store, cannot be undone once launched. So an instruction is considered to have completed before an external interrupt is taken.

### Exception Handling

- When instruction $x$ in stage $i$ raises an exception, its cause is recorded and passed down the pipeline.
- For a given instruction, exceptions from the later stages of the pipeline do not override cause of exception from the earlier stages.
- If an exception is present at commit: Cause and EPC registers are set, and pc is redirected to the handler PC.
  - Epoch mechanism takes care of redirecting the pc.
Killing vs Poisoning

This affects whether an instruction is removed from sb in case of an interrupt.

Interrupt processing at Execute

- if (mem type) issue Ld/St
- if (mispred) redirect
- pass eInst to M stage

eInst will contain information about any newly detected interrupts at Execute.
Interrupt processing at Memory stage

Incoming Interrupt:
- no
- yes

Memory Interrupt?
- no
- yes

- pass eInst with modified data to Commit
- pass eInst to Commit unmodified
- pass new Cause to Commit

Interrupt processing at Commit

External Interrupt?
- no
- yes

Incoming interrupt:
- no
- yes

- commit
- sb.rm

EPC<= pc;
causeR <= inCause;
if (inCause after Reg Fetch) sb.rm;
mode <= privilege;
Redirect

EPC<= pc;
causeR <= Ext;
mode <= privilege;
Redirect
Final comment

- There is generally a lot of machinery associated with a plethora of exceptions in ISAs
- Precise exceptions are difficult to implement correctly in pipelined machines
- Performance is usually not the issue and therefore sometimes exceptions are implemented using microcode

RISC-V Virtual Memory
Privileged ISA v. 1.9.1
RISC-V Privilege Levels

- Separation between low-level access to the hardware and high-level user programs
  - Machine-mode (M) – all addresses are physical addresses, has access to all addresses including memory-mapped IO devices
  - Supervisor-mode (S) – addresses are typically virtual addresses, can switch page-table in use (sptbr)
  - User-mode (U) – addresses are virtual, access to devices only through systemcalls

RISC-V Memory Maps

- Demand Paging makes the entire address space look like DRAM
- Only part of the address space is DRAM
RISC-V Paged Virtual Memory

Different modes for different systems:

- **Sv32**: 32-bit VA, 34-bit PA
  - 4 GB virtual address space
  - 16 GB physical address space
  - 2-layer page table
- **Sv39**: 39-bit VA, 50-bit PA
  - 512 GB virtual address space
  - 1 PB physical address space
  - 3-layer page table

November 13, 2015

http://csg.csail.mit.edu/6.175

---

**Sv32 Addresses**

Virtual Addresses:

- **VPN[1]**: 10 bits
- **VPN[0]**: 10 bits
- **page offset**: 12 bits

1st level Page Table Index 2nd level Page Table Index

Physical Addresses:

- **PPN[1]**: 12 bits
- **VPN[0]**: 10 bits
- **page offset**: 12 bits

Both come from 2nd level Page Table Entry

November 13, 2015

http://csg.csail.mit.edu/6.175
### Sv32 Page Table Entries

<table>
<thead>
<tr>
<th>PPN[1]</th>
<th>PPN[0]</th>
<th>SW Reserved</th>
<th>D</th>
<th>A</th>
<th>G</th>
<th>U</th>
<th>X</th>
<th>W</th>
<th>R</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 bits</td>
<td>10 bits</td>
<td>2 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Dirty** – This page has been written to  
**Accessed** – This page has been accessed  
**Global** – Mapping exists in *all* virtual address spaces  
**User** – User-mode programs can access this page  
**eXecute** – This page can be executed  
**Write** – This page can be written to  
**Read** – This page can be read from  
**Valid** – This page valid and in memory

If \( V = 1 \), but \( X, W, R = 0 \), \( \text{PPN}[\] points to the 2\(^{nd}\) level page table

If \( V = 0 \), page is either invalid or in disk. If in disk, the OS can reuse bits in the PTE to store the disk address (or part of it).

<table>
<thead>
<tr>
<th>Disk Address</th>
<th>G</th>
<th>U</th>
<th>X</th>
<th>W</th>
<th>R</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>26 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

November 13, 2015  
http://csg.csail.mit.edu/6.175
RISC-V Pipeline with VM

On a fault, an exception is raised and the OS takes over.

RISC-V VM Instructions

- **SFence.VM**
  - Privileged instruction to synchronize TLB translation. Ensures that stores to data cache are seen by hardware page table walker.

- **CSRs – Control and Status Registers**
  - Privileged registers for processor configuration
  - **sptbr** – Page Table Base Register
  - **mstatus.vm** – Virtual Memory mode (e.g. Sv32)
  - **mstatus.mxr, mstatus.pum, mstatus.mprv** – Fields for modifying privileges for memory accesses to emulate accesses at low privilege levels.