Learning Outcomes

• An understanding of virtual linear array page tables, and their use on the MIPS R3000.
• Exposure to alternative page table structures beyond multi-level and inverted page tables.

Virtual Linear Array page table

• Uses a page table array index by page number
• Page table array is in virtual memory with only used parts of the array allocated in physical memory
• A second page table root node has translations for the page table itself

Virtual Linear Array Operation

• Index into page table array without referring to root PT
• Simply use the full page number as the PT index
• Leave unused parts of PT unmapped
• If access is attempted to unmapped part of PT, a secondary page fault is triggered
  – This will load the mapping for the PT from the root PT
  – Root PT is kept in physical memory (cannot trigger page faults)
R3000 TLB Refill

- Dedicated exception handler
- Can be optimised for TLB refill only
  - Does not need to check the exception type
  - Does not need to save any registers
    - It uses a specialised assembly routine that only uses k0 and k1.
    - Does not check if PTE exists
    - Assumes virtual linear array – see extended OS notes
  - With careful data structure choice, exception handler can be made very fast
- An example routine
  mfc0 k1,C0_CONTEXT
  mfc0 k0,C0_EPC # mfc0 delay slot
  lw k1,0(k1) # may double fault (k0 = orig EPC)
  nop
  mfc0 k1,C0_EPT
  tlbwr
  jr k0
  rfe

How does this work?

c0 Context Register

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| PTEBase | Ebd VPN | 0 |

- c0_Context = PTEBase + 4 * PageNumber
  - PTEs are 4 bytes
  - PTEBase is the base local of the page table array (note: aligned on 4 MB boundary)
  - PTEBase is (re)initialised by the OS whenever the page table array is changed
    - E.g. on a context switch
  - After an exception, c0_Context contains the address of the PTE required to refill the TLB.

Virtual Linear Array Page Table

- Use Context register to simply load PTE by indexing a PTE array in virtual memory
- Occasionally, will get double faults
  - A TLB miss, while servicing a TLB miss
  - Handled by general exception handler
- Protected from user access

Virtual Linear Array Page Table

Code for VLA TLB refill handler

- Load PTE address from context register
  mfc0 k1,C0_CONTEXT
  mfc0 k0,C0_EPC
  lw k1,0(k1)
  nop
  mfc0 k1,C0_EPT
  tlbwr
  jr k0
  rfe

- Move the PTE into EntryLo.

- Write EntryLo into random TLB entry.

- Return from the exception

Load address of instruction to return to

Note: this load can cause a TLB refill miss itself, but this miss is handled by the general exception vector. The general exception vector has to understand this situation and deal with it appropriately.
Software-loaded TLB

- Pros
  - Can simplify hardware design
  - provide greater flexibility in page table structure
- Cons
  - typically have slower refill times than hardware managed TLBs.

Design Tradeoffs for Software-Managed TLBs
David Nagle, Richard Uhlig, Tim Stanley, Stuart Sechrest, Trevor Mudge & Richard Brown
ISCA '93 Proceedings of the 20th annual international symposium on computer architecture

Trends at the time

- Operating systems
  - moving functionality into user processes
  - making greater use of virtual memory for mapping data structures held within the kernel.
- RAM is increasing
  - TLB capacity is relatively static
- Statement:
  - Trends place greater stress upon the TLB by increasing miss rates and hence, decreasing overall system performance.
  - True/False? How to evaluate?

Software Trap on TLB Miss

Figure 1: Tapeworm

The Tapeworm TLB simulator is built into the operating system and is invoked whenever there is a real TLB miss. The simulator uses the real TLB miss to simulate its own TLB configuration(s). Because the simulator resides in the operating system, Tapeworm captures the dynamic nature of the system and avoids the problems associated with simulators driven by static traces.

Figure 2: Page Table Structure in OSIF/1 and Mach 3.0

The Mach page table is a linear address. The bits in the lower 27 bits of the linear address are used to determine the location of the page table entry. The upper 5 bits of the linear address are used to index into the page table. Each page table entry contains two pointers, one for the page table and one for the page frame. The page table entry is stored in the physical page frame pointed to by the lower pointer. The page frame is a 4KB block of physical memory. The page table entry contains a bit for each page frame in the page table to indicate whether or not the page frame is present in memory. The page table entry is used to map a linear address to a physical address. The linear address is divided into a page number and an offset. The page number is used to index into the page table to get the physical address of the page frame containing the page. The offset is used to map the page frame to a physical address within the page frame.

Figure 3: Costs for Different TLB Miss Types

This table shows the number of machine cycles (at 60 MIPS) required to service different types of TLB misses. The simulator was used to collect a 120M-word database of thrashing for each type of miss. The simulator is a 3-level cache with the miss characteristics described below. Note that Mach does not have L3 misses because it implements a 2-level page table.

<table>
<thead>
<tr>
<th>TLB Miss Type</th>
<th>Unix</th>
<th>OSIF/1</th>
<th>Mach 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1U</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>L1K</td>
<td>200</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>L2</td>
<td>400</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>L3</td>
<td>500</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Invalid</td>
<td>500</td>
<td>500</td>
<td>500</td>
</tr>
</tbody>
</table>

Table 3: Costs for Different TLB Miss Types
Note the TLB miss costs

- What is expected to be the common case?

Measurement Results

<table>
<thead>
<tr>
<th>System</th>
<th>Total Time (ms)</th>
<th>L1/K</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>Invalid</th>
<th>Misses</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix</td>
<td>300</td>
<td>90</td>
<td>120</td>
<td>720</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>1040</td>
</tr>
<tr>
<td>OSF1</td>
<td>300</td>
<td>90</td>
<td>120</td>
<td>720</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>1040</td>
</tr>
<tr>
<td>Mach 3.0</td>
<td>300</td>
<td>90</td>
<td>120</td>
<td>720</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>1040</td>
</tr>
</tbody>
</table>

Table 1: Number of TLB Misses

<table>
<thead>
<tr>
<th>System</th>
<th>Total TLB Service Time (ms)</th>
<th>L1/K</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>Invalid</th>
<th>Misses</th>
<th>% of Total Misses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix</td>
<td>300</td>
<td>90</td>
<td>120</td>
<td>720</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>0.26%</td>
</tr>
<tr>
<td>OSF1</td>
<td>300</td>
<td>90</td>
<td>120</td>
<td>720</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>0.26%</td>
</tr>
<tr>
<td>Mach 3.0</td>
<td>300</td>
<td>90</td>
<td>120</td>
<td>720</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>0.26%</td>
</tr>
</tbody>
</table>

Table 2: Time Spent Handling TLB Misses

Specialising the L2/L1K miss vector

<table>
<thead>
<tr>
<th>Type of TLB Misses</th>
<th>Counts</th>
<th>Predicted Cost (ms)</th>
<th>New Total Cost (ms)</th>
<th>Time Saved (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>30,045</td>
<td>10,000</td>
<td>10,000</td>
<td>0.00</td>
</tr>
<tr>
<td>L2</td>
<td>30,045</td>
<td>10,000</td>
<td>10,000</td>
<td>0.00</td>
</tr>
<tr>
<td>L3</td>
<td>30,045</td>
<td>10,000</td>
<td>10,000</td>
<td>0.00</td>
</tr>
<tr>
<td>Invalid</td>
<td>5,000</td>
<td>5,000</td>
<td>5,000</td>
<td>0.00</td>
</tr>
<tr>
<td>Total</td>
<td>35,045</td>
<td>10,000</td>
<td>10,000</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 3: Reconfigured Cost of TLB Misses Given Additional Miss Vectors (Mach 3.0)
Itanium Page Table

- Takes a bet each way
- Loading
  - software
  - two different format hardware walkers
- Page table
  - software defined
  - Virtual linear array
  - Hashed

That is it!