R3000 TLB Refill

- 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, COCONTEXT
  mfc0 k0, CO_EPC # mfc0 delay
  lw k1, 0(k1) # may double fault (k0 = orig EPC)
  nop
  mtc0 k1, CO_ENTRYLO
  nop
  tlbwr
  jr k0
  rfe
  ```

Virtual Linear Array page table

- Assume a 2-level PT
- Assume 2nd-level PT nodes are in virtual memory
- Assume all 2nd-level nodes are allocated contiguously ⇒ 2nd-level nodes form a contiguous array indexed by page number

Virtual Linear Array Operation

- Index into 2nd-level page table 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)

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

PTEbase in virtual memory in kseg2
- Protected from user access
c0 Context Register

- \( c0_{\text{Context}} = \text{PTEBase} + 4 \times \text{PageNumber} \)
  - PTEs are 4 bytes
  - PTEBase is the base local of the page table array (note: aligned on 4 MB boundary)
  - PTEBase is (re)initiated by the OS whenever the page table array is changed
  - E.g., on a context switch
  - After an exception, \( c0_{\text{Context}} \) contains the address of the PTE required to refill the TLB.

Code for VLA TLB refill handler

1. Load PTE address from context register
2. Move the PTE into EntryLo
3. Write EntryLo into random TLB entry
4. Load address of instruction to return to
5. Return from the exception

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.