Translation Lookaside Buffer

Virtual Memory would not be very effective if every memory address had to be translated by looking up the associated physical page in memory. The solution is to cache the recent translations in a Translation Lookaside Buffer (TLB)

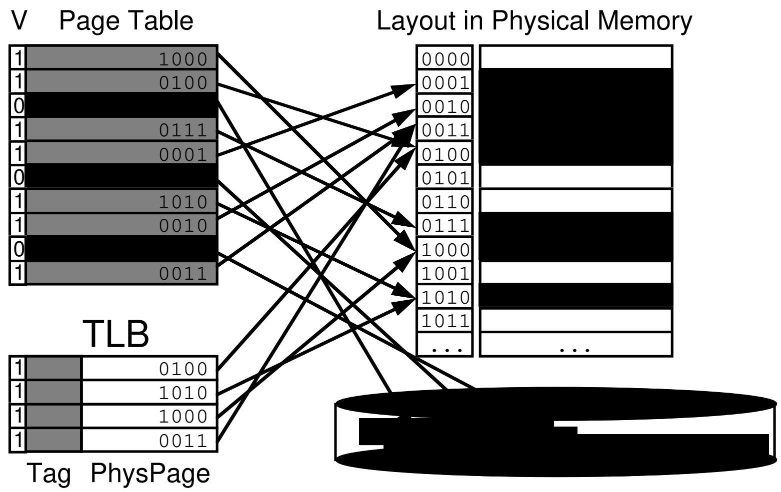
Addressing The Cache

Since the program is generating virtual addresses, and the memory uses physical addresses, there are two solutions to the problem of addressing the cache:

- Physically addressed cache: Translate virtual address before cache reference
- Virtually addressed cache: Reference cache directly and translate only on a cache miss, which is when physical memory must be referenced

It’s a no brainer, right?

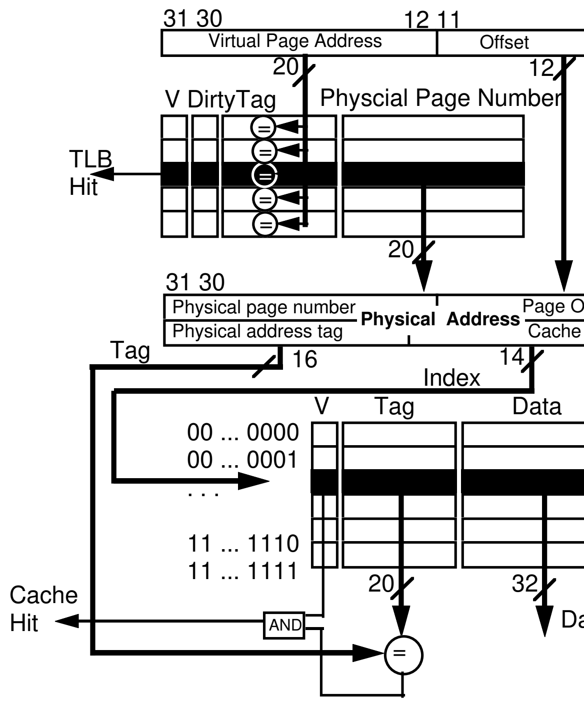
Translation Lookaside Buffer



Translation Lookaside Buffer

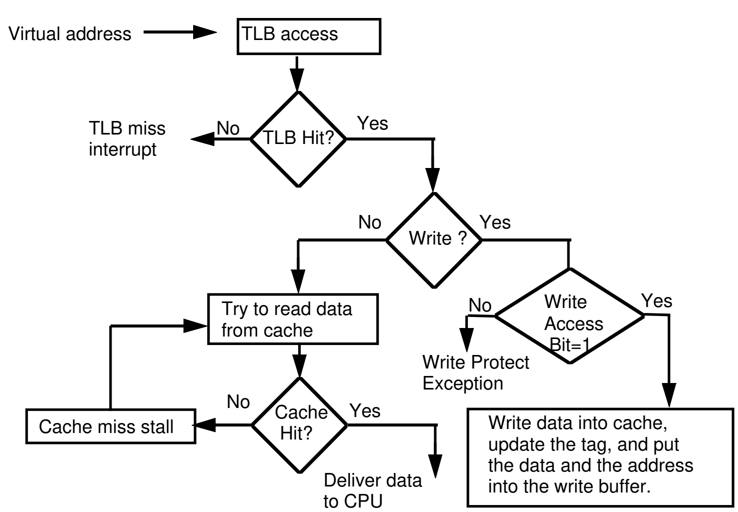
To save time in virtual to physical address translation, temporal locality is exploited by keeping a small cache of the most recent virtual-physical mappings.

Block size	1-2 page-table entries
Hit Time	1/2-1 clock cycle
Miss penalty	10-30 clock cycles
Miss rate	0.01%-1%
Size	32-1024 entries



DEC 3100
Cache

Logic of Memory Reference for 3100



TLB Miss Means Either ...

- The page is present ==> only a TLB entry must be created.
- The page is not present (i.e. page table entry for the virtual address has 0 valid bit), a page fault exception is signalled.
    - The exception flushed the instruction, put the PC in the exception program counter (EPC) and interrupted the processor.
    - The operating systems, checking the cause, discovers a page fault was signalled, and knowing this is a time consuming operation, saves the state: GP and FP registers, Page Table Address, EPC & Cause.
    - What address is needed:
        - Instruction Page Fault, find address in EPC.
        - Data Page Fault, compute address from Inst.
    - OS then:
        - Finds disk address in page table entry.
        - Chooses victim to replace; writes back if dirty bit set.
        - Initiates read of disk block.

Protection Through Virtual Memory

A multi-user environment requires protection.

Virtual address spaces are logically separate as long as they never reference the same physical page.

Operating System sets page tables.

Two execution modes: user/supervisor.

Page table address must be supervisor readable.

Sharing can be assisted with "write protection" or read/write bits.

Context switching can be assisted when there is a TLB by extending the tag field of TLB entry with a process ID.

Matches require both the address and the ID to match.

Exercises

Memory:

Address	Contents
000000ac	
000000b0	
000000b4	
. . .	
000080ac	
000080b0	
000080b4	
. . .	
0000e0ac	80000000
0000e0b0	8000000e
0000e0b4	8000a0b4

Page Table Address: 0000e0a8

Assuming 4K pages and "big-endian" addressing, i.e. the 0 byte of a word is the msb end, what are the contents of the memory location at the virtual address 000020b7?

The TLB physical page number for tag 00002 would be?

Assuming 4K pages and assuming the physical address from the TLB is 000000b4, what is the tag field?

Solutions

Memory:

Address	Contents
000000ac	
000000b0	
000000b4	
. . .	
000080ac	
000080b0	
000080b4	
. . .	
0000e0ac	80000000
0000e0b0	8000000e
0000e0b4	8000a0b4

Page Table Address: 0000e0a8

Q. Assuming 4K pages and "big-endian" addressing, i.e. the 0 byte of a word is the msb end, what are the contents of the memory location at the virtual address 000020b7?

A. "0b7" is the page offset, and "00002" is the virtual page number. Convert this to a displacement for the page table by multiplying by 4, and add to the page table address: 0000e0a8 + 8 = 0000e0b0". Find the physical page number at that location: 8000000e, where the msb is the "valid bit". Construct the physical address: 0000e0b7. Find that byte in memory: b4.

Solutions

Memory:

Address	Contents
000000ac	
000000b0	
000000b4	
. . .	
000080ac	
000080b0	
000080b4	
. . .	
0000e0ac	80000000
0000e0b0	8000000e
0000e0b4	8000a0b4

Page Table Address: 0000e0a8

Q. The TLB physical page number for tag 00002 would be?

A. The TLB is a cache, so it has a tag field. The tag field holds the (portion of the) virtual page number (00002) corresponding to the physcial page number. The previous question determined the physcial page number corresponding to this virtual page number: 0000e.

Solutions

Memory:

Address	Contents
000000ac	
000000b0	
000000b4	
. . .	
000080ac	
000080b0	
000080b4	
. . .	
0000e0ac	80000000
0000e0b0	8000000e
0000e0b4	8000a0b4

Page Table Address: 0000e0a8

Q. Assuming 4K pages and assuming the physical address from the TLB is 000000b4, what is the tag field?

A. "Physcial address from the TLB" means the physical address produced using the TLB. Since the page offset is 0b4, the physical page number is 00000. Which page table entry maps to that physical page number? The page table entry at 0...0e0ac. Subtracting from this the page table address yields: 0000e0ac - 0000e0a8 = 4. This is an address, which by dividing by 4, we can convert to a virtual page number: 4/4 = 1. So, 00001 must be in the tag field.