zachary_hightower_hwk_9

CSCI 223 Computer Organization & Assembly Language

Homework #9

Due: 11:59pm Monday 4/8/2024
Submission: type your answer electronically and upload it (e.g., a doc, pdf, odt, or txt file) to the Blackboard

Simulate a small virtual memory system with TLB and L1 D-cache with following assumptions:

- The memory is byte addressable
- Memory accesses are to 1-byte words (not 4-byte words)
- Virtual addresses are 14 bits wide (n = 14)
- Physical addresses are 12 bits wide (m = 12)
- The page size is 64 bytes (P = 64)
- The TLB is four-way set associative with 16 total entries
- The L1 D-cache is physically addressed and direct-mapped, with a 4-byte line size and 16 total sets

Figure below shows the format of the virtual and physical addresses. Since each page is 2^6 = 64 bytes, the low-order 6 bits of the virtual and physical addresses serve as the VPO and PPO respectively. The high-order 8 bits of the virtual address serve as the VPN. The high-order 6 bits of the physical address serve as the PPN.

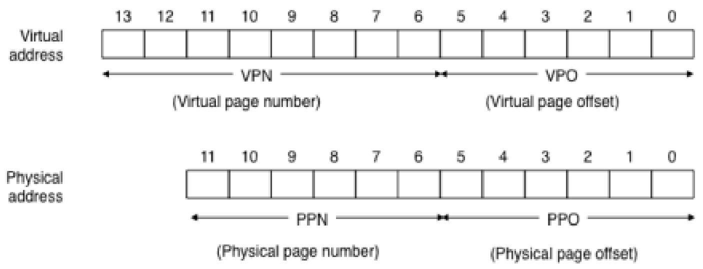
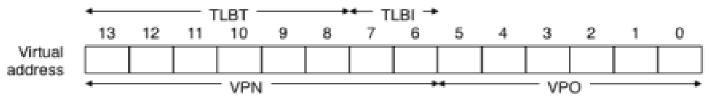
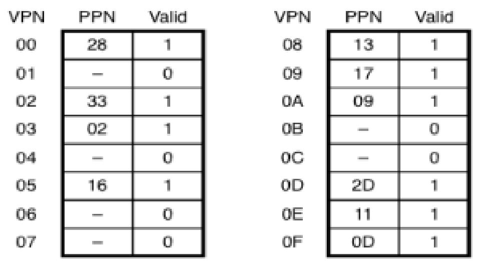


Figure below shows a snapshot of our little memory system including the TLB (a), a portion of the page table (b), and the L1 cache (c). Above the figures of the TLB and cache, we have also shown how the bits of the virtual and physical addresses are partitioned by the hardware as it accesses these devices.



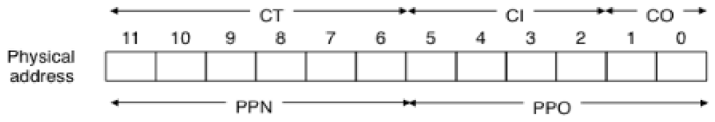
(a) TLB: Four sets, 16 entries, four-way set associative

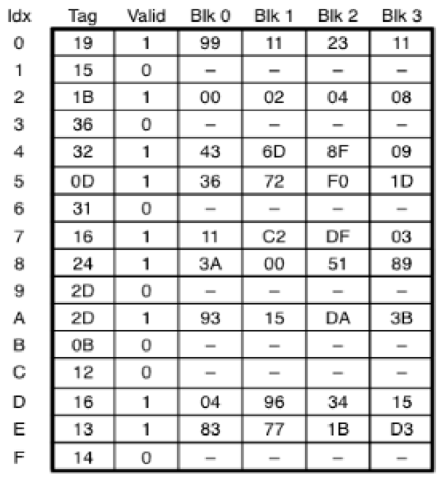
Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0



VPN	PPN	Valid	VPN	PPN	Valid
00	28	1	08	13	1
01	–	0	09	17	1
02	33	1	0A	09	1
03	02	1	0B	–	0
04	–	0	0C	–	0
05	16	1	0D	2D	1
06	–	0	0E	11	1
07	–	0	0F	0D	1

(b) Page table: Only the first 16 PTEs are shown





Idx	Tag	Valid	Blk 0	Blk 1	Blk 2	Blk 3
0	19	1	99	11	23	11
1	15	0	–	–	–	–
2	1B	1	00	02	04	08
3	36	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	–	–	–	–
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–

(c) Cache: Sixteen sets, 4-byte blocks, direct-mapped

- TLB: The TLB is virtually addressed using the bits of the VPN. Since the TLB has four sets, the 2 low-order bits of the VPN serve as the set index (TLBI). The remaining 6 high-order bits serve as the tag (TLBT) that distinguishes the different VPNs that might map to the same TLB set.
- Page table: The page table is a single-level design with a total of $2^8 = 256$ page table entries (PTEs). However, we are only interested in the first sixteen of these. For convenience, we have labeled each PTE with the VPN that indexes it: but keep in mind that these VPNs are not part of the page table and not stored in memory. Also notice that the PPN of each invalid PTE is denoted with a dash to reinforce the idea that whatever bit values might happen to be stored there are not meaningful.
- Cache: The direct-mapped cache is addressed by the fields in the physical address. Since each block is 4 bytes, the low-order 2 bits of the physical address serve as the block offset (CO). Since there are 16 sets, the next 4 bits serve as the set index (CI). The remaining 6 bits serve as the tag (CT).

Given this initial setup, explain (by simulating manually) what happens in order when the CPU executes a load instruction that reads the byte at address **0x03d4**.

We want to simulate what happens when we run a load instruction meant to read the byte at address 0x03d4 with our outlined setup above.

0000 0011 1101 0100 = 0x03d4

Step 1: Virtual Address

We need to split apart the given address into its component pieces in binary, so we can use it properly.

- VPN = 0x0F = 00001111
- VPO = 0x14 = 010100
  Now we have the two parts necessary to perform the next step

Step 2: TLB

- The VPN tells us the index to look for according to the two low order bits, 11 = 3
- Then we need to know the tag to look for, which is also 03 = 000011
- Then, we search through the set index for the tag, we find it in the second group
- Check the PPN. Our PPN = 0D
- Check that it's valid, we can see that the slot is occupied by a 1, so it is valid

Step 3: Page Table

- We got a hit in the TLB, so we can ignore the page table

Step 4: Physical Address

- We want to combine the PPN we've found (0x0D), with the VPO (0x14) to make the full physical address
- Find the PPO, Physical Page Offset. It's equal to the VPO = 0x14 = 010100
- So our physical address is 0x0D + 0x14 = 0x0D14 = 0000 0000 1101 0001 0100

Step 5: Cache

- We need the tag, index, and offset for the cache
- The cache tag is 00110100 = 0x34
- The cache index is 0101 = 0x5
- The cache offset is = 00 = 0x0
- We go to index 5 in our cache
- We look for our tag (0x34)
- We find the incorrect tag, so we state that this is a miss at the cache level

Step 6: Byte

- We do not extract any byte from the operation due to the miss

OR

Step 5: Cache

- We need the tag, index, and offset for the cache
- The cache tag ***is meant to be read as*** 0x0D 0000 1101 = 0x0D
- The cache index is 0101 = 0x5
- The cache offset is = 00 = 0x0
- We go to index 5 in our cache
- We look for our tag (0x0D)
- We find the correct tag, and the valid bit is set, so this is a hit in the cache
- We look through the bytes and find the one at offset 0, which is 36

Step 6: Byte

- We extract byte 36 and the operation returns it