LE16.2

LE 16.2.1: Page faults galore!

0.0/1.0 point (ungraded)

1. A particular Beta implementation has 32-bit virtual addresses, 32-bit physical addresses and a page size of 2^{12} bytes. A test program has been running on this Beta and has been halted *just before* execution of the following instruction at location 0×1FFC:

```
LD(R31,0x34C8,R1) | PC = 0x1FFC
ST(R1,0x6004,R31) | PC = 0x2000
```

The first 8 locations of the page table at the time execution was halted are shown below; the least recently used page (“LRU”) and next least recently used page (“next LRU”) are as indicated. Assume that all the pages in physical memory are in use. Execution resumes and the LD and ST instructions are executed.

VPN	D	R	PPN
0	1	1	0×1
1	0	1	0×0
LRU→2	1	1	0×6
3	--	0	--
Next LRU→4	0	1	0×4
5	0	1	0×2
6	0	1	0×7
7	0	1	0×3

Please **fill in the contents of the page table** after the ST instruction has completed execution. For resident pages set R = 1 and indicate whether the dirty (D) bit is a 0 or a 1. For the PPN enter the physical page number in hex. **If a virtual page is removed from memory, indicate that by setting R = 0 and setting the dirty bit and the PPN to "--"**.

VPN	D	R	PPN
0	Select an option Answer: 1	Select an option Answer: 1	0x: Answer: 1
1	Select an option Answer: 0	Select an option Answer: 1	0x: Answer: 0
2	Select an option Answer: 0	Select an option Answer: 1	0x: Answer: 4
3	Select an option Answer: 0	Select an option Answer: 1	0x: Answer: 6
4	Select an option Answer: --	Select an option Answer: 0	0x: Answer: --
5	Select an option Answer: 0	Select an option Answer: 1	0x: Answer: 2
6	Select an option Answer: 1	Select an option Answer: 1	0x: Answer: 7
7	Select an option Answer: 0	Select an option Answer: 1	0x: Answer: 3

Calculator

Answer: 3

Explanation

We are told that the size of a page is 2^{12} bytes. This means that the bottom 3 hex characters are used to indicate the page offset, the top hex character indicates the virtual page number being referenced. For the load instruction, whose address is 0×1FFC, that means that virtual page 1 is referenced. Then the data address that is fetched within the LD is 0×34C8 which is in VPN 3. Similarly, for the store instruction, whose address is 0×2000, virtual page 2 is referenced. Then the data address used by the store instruction is 0×6004 which is in VPN 6.

When trying to access virtual page 1, we find that the page is resident in the physical memory at PPN 0. We then try to access VPN 3 and get a miss. This causes us to remove the LRU page from physical memory so that we can put VPN 3 in its place. Note that because the dirty bit of VPN 2 is set, that means that we must first write PPN 6 back to disk before using PPN 6 for VPN 3. After writing the page back to disk, we mark VPN 2 as non-resident by setting R = 0. The values of D and PPN are ignored when R = 0. Now, we can use PPN 6 for VPN 3. This is indicated in the page map by setting R = 1 and PPN = 6 for VPN 3. The dirty bit is set to 0 because the physical page has not been changed since it was brought in from disk.

So our updated page table after the completion of the LD instruction looks like this:

VPN	D	R	PPN
0	1	1	0×1
1	0	1	0×0
LRU→2	--	0	--
3	0	1	0×6
Next LRU→4	0	1	0×4
5	0	1	0×2
6	0	1	0×7
7	0	1	0×3

Next, the ST instruction is fetched from VPN 2. This causes a miss since VPN 2 was removed from physical memory to make room for VPN 3. So we need to bring VPN 2 back into physical memory by removing the next LRU VPN which is 4. This changes the resident bit of VPN 4 to 0, and now PPN 4 which was previously used by VPN 4 is available for VPN 2. So the resident bit of VPN 2 is set to 1, and the PPN is set to 0×4. Finally, we want to write to VPN 6. This page is already resident in physical memory, so the only change that occurs in our page table is that the dirty bit for VPN 6 is set to 1 to indicate that we have modified the page and physical memory and it must be copied back to disk before being removed from physical memory.

So our final page table after the completion of the LD and ST instructions looks like this:

VPN	D	R	PPN
0	1	1	0×1
1	0	1	0×0
LRU→2	0	1	0×4
3	0	1	0×6
Next LRU→4	--	0	--
5	0	1	0×2
6	1	1	0×7
7	0	1	0×3

2. Which physical pages, if any, needed to be written to disk during the execution of the LD and ST instructions?

Physical page numbers written to disk or NONE:

- [ ] 0×0
- [ ] 0×1
- [ ] 0×2
- [ ] 0×3
- [ ] 0×4

Calculator

- [ ] 0×5
- [x] 0×6
- [ ] 0×7
- [ ] NONE

Explanation
When VPN 2 is removed from physical memory to make room for VPN 3, VPN 2 which is currently in PPN 6, must be written back to disk because the dirty bit for VPN 2 was equal to 1 meaning that the page had been changed in physical memory and is out of sync with the copy on disk.

3. Please give the 32-bit physical memory addresses used for the four memory accesses associated with the execution of the LD and ST instruction.

32-bit physical memory address of LD instruction: 0x _____ Answer: 0FFC

32-bit physical memory address of data read by LD: 0x _____ Answer: 64C8

32-bit physical memory address of ST instruction: 0x _____ Answer: 4000

32-bit physical memory address of data written by ST: 0x _____ Answer: 7004

Explanation
The virtual address for the load instruction was 0×1FFC. We saw that this means that it comes from VPN 1 which mapped to PPN 0, so the physical address of the LD instruction is 0×0FFC. The data that is read by the load is at virtual address 0×34C8. Here VPN 3 is in PPN 6, so the physical address of the data read by the LD is 0×64C8.
The virtual address of the store instruction is 0×2000. Since VPN 2 is stored at PPN 4, this means that the store instruction came from physical address 0×4000. The location accessed by the store instruction is virtual address 0×6004. Since VPN 6 maps to PPN 7, the data written by the store is written to physical address 0×7004.

‹ Previous Next ›

Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap
Cookie Policy
Your Privacy Choices

Connect

Idea Hub
Contact Us
Help Center
Security
Media Kit

Calculator