7. Virtual Memory [80 points]

We have added a 4-set, 2-way write-through physically-indexed physically-tagged cache and an x86 style 2-level virtual-to-physical translation scheme to the LC-3b. Each virtual address is 16 bits. VA[15:11] is used to index the page directory, VA[10:6] is used to access the level 2 page table, and VA[5:0] is used to access the byte in the page. The PDE/PTE (page directory entry/page table entry) only contains a valid bit as its most significant bit and the PFN (page frame number) as its least significant bits. All remaining bits in the PDE/PTE are reserved and always set to zeros.

Assume the following:

- Physical address is 10 bits
- The cache implements a perfect LRU replacement policy
- The cache block size is 4 bytes
- Instructions and data share the same cache
- The cache is initially empty
- The page directory or a page table is one page in size
- Memory accesses to the page directory, page tables, and pages are **all** cached
- No page faults happen during the execution of an instruction
- Assume that the LC-3b does **not** permit unaligned accesses
- If the LRU bit is 0, the data in Way 0 will be evicted next
- For decoding instructions and page table/directory entries, the byte at the lower address is at bits [15:8] and the byte at the higher address is at bits [7:0]. (i.e., we assume **big endian**)

The PC is loaded with address x3184 and one instruction is run to completion. The figure below shows the contents of the cache **after** the execution of this instruction is complete. For each cache block, we index the data from left to right (e.g., in $Data_0$ of Set_1, if byte a references $0x60$, byte $a+1$ references $0x43$).

Use the figure below to answer the questions on the following three pages. There is an LC-3b ISA cheatsheet for your benefit in the appendix.

	LRU	V_0	Tag_0	V_1	Tag_1	$Data_0$	$Data_1$
Set_0	0	0	101100	0	010010	x80 x70 xAE xFD	x80 x06 x80 x77
Set_1	1	1	011100	0	011100	x60 x43 x59 xAE	xFD xE9 x46 x57
Set_2	0	0	011000	0	110110	x53 x74 x65 x70	x68 x65 x6E x21
Set_3	0	1	111100	1	110000	x80 x0C x00 x0F	x80 x07 x80 x06

Initials:

(a) How big is each page (in bytes)? Show your work.

2^6

(b) What is the value in the Page Directory Base Register? Show your work.

x3C0

(c) Fill in the table in the order of physical memory accesses caused by the instruction. Note that some rows in this table may be left blank.

Physical Address	**Data**	**Explanation of Data**
x3CC	x800C	PDE of instruction
x30C	x8007	PTE of instruction
x1C4	x6043	Instruction
x3CC	x800C	PDE of Data
x30C	x8007	PTE of Data
x1C4 or x1C6	x6043 or x59AE	Data

Show your work in the box below.

	LRU	V_0	Tag_0	V_1	Tag_1	$Data_0$	$Data_1$
Set_0	0	~~0~~	~~101100~~	~~0~~	~~010010~~	~~x80 x76 xAE xFD~~	~~x80 x06 x80 x77~~
Set_1	1	1	011100	~~0~~	~~011100~~	x60 x43 x59 xAE	~~xFD xE9 x40 x57~~
Set_2	0	~~0~~	~~011000~~	~~0~~	~~110110~~	~~x53 x74 x65 x70~~	~~x68 x65 x6E x21~~
Set_3	0	1	111100	1	110000	x80 x0C x00 x0F	x80 x07 x80 x06

- We ignore the entries crossed out in red because they are not valid entries in the cache.
- There are 3 valid entries in the cache. One must hold the instruction, one must hold the page table entry, and one must hold the page directory entry.
- The values underlined in green look like PDE/PTEs. (there is a valid bit at the MSB of each)
- The cache blocks in Set3 must contain the PTE and PDE and they cannot be in the same cache block.
- Looking at the LRU bit (due to the nature of page table walking), we can determine that x800C is the PDE and the PTE is either x8007 or x8006.
- We notice that all interesting data is in sizes of two bytes and aligned.
- From the cache we can find the addresses associated with each piece of data. Mem[x1C4] = x6043 Mem[x1C6] = x59AE Mem[x3CC] = x800C Mem[x30C] = x8007 Mem[x30E] = x8006

0x3184 is the address of the instruction being executed. Below is the binary representation of the address.

16h'3184 = 16b'00110_00110_000100

PD Index: 6 PT Index: 6 Page Index: 4

We see that the instruction must have a physical address with the following constraint: xxxx000100, because of the page index. Mem[x1C4] = x6043 is the only entry in the cache that follows this rule.
Now we know that x6043 is the instruction (i.e., a load instruction which is decoded into R0 = Mem[R1 + 6]). Since pages are aligned, we know that the page containing the two possible PTEs starts at address x300: Mem[x30C] = x8007 and Mem[x30E] = x8006. Because the page table index is 6, we know that x8007 is the PTE. We must multiply the index by two because the size of each entry is 2 bytes. We have determined which entries in the cache are the PTE, PDE and the instruction.

(d) What are the values of R0 and R1? There are two possible answers to this question and either one will get full credit. Show your work.

R0: x6043 or x59AE R1: x317E or x3180
Since the instruction that was executed was a load, we must walk through the page tables once again with the unknown address (R1 + 6). Given that there are no other entries valid in the cache, we can infer that the this address hits the same page. By this logic, either 0x6043 or x59AE is the data that was loaded into R0. Depending on which data value you chose, the address in R1 will be either x317E or x3180 (remember we added 6 to this address before probing memory).