EX for Lecture 11 Virtual Memory

1. A system implements a paged virtual address space for each process using a one-level page table. The maximum size of virtual address space is 16MB. The page table for the running process includes the following valid entries (the → notation indicates that a virtual page maps to the given page frame, that is, it is located in that frame):

Virtual page 2 → page frame 4	Virtual page 4 → page frame 9
Virtual page 1 → page frame 2	Virtual page 3 → page frame 0
Virtual page 0 → page frame 1	

The page size is 1024 bytes and the maximum physical memory size of the machine is 2MB.
a. How many bits are required for each virtual address?
b. How many bits are required for each physical address?
c. What is the maximum number of entries in a page table?
d. To which physical address will the virtual address 1524_{10} translate?
e. Which virtual address will translate to physical address 1023_{10}?

Ans.
a. There are 16MB, or $2^4 x 2^{20}$ addresses, so we need 24 bits for a virtual address.

b. Main memory is 2MB, or $2^1 x 2^{20}$, so we need 21 bits for a physical address.

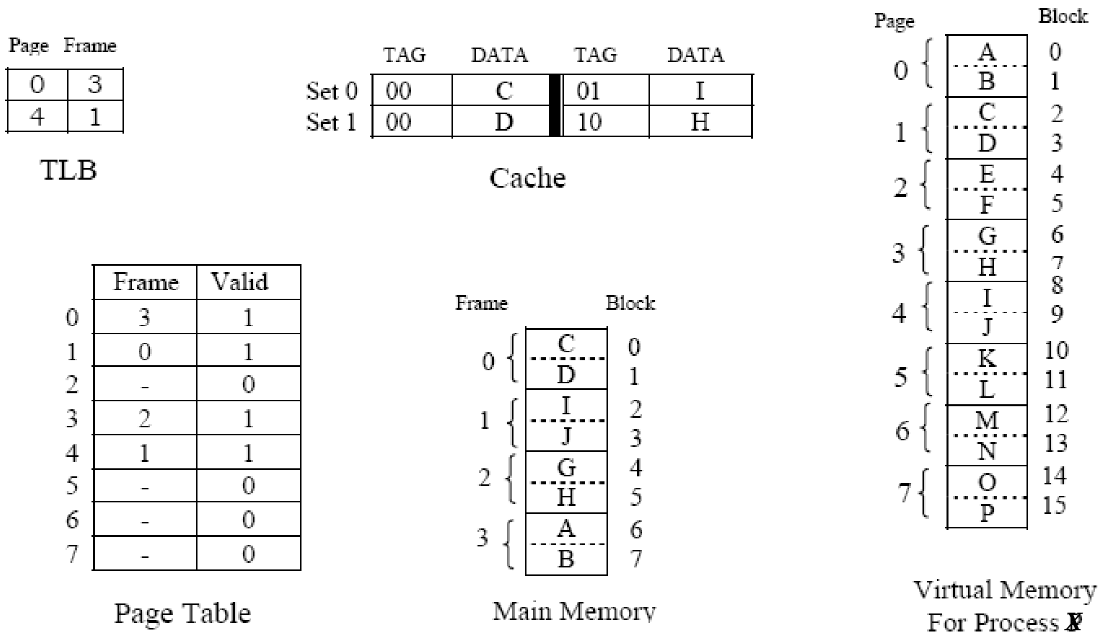
c. There are $2^{24}/2^{10}$ pages in virtual memory, so the page table can have 2^{14} entries.

d. 1524 is on page 1 (page 0 contains addresses 0-1023, page 1 contains 1024 - 2047), located at offset 500. Page 1 maps to frame 2, so 1524 maps to physical address 2548. You can also find the binary equivalent of 1524 (000000000000010111110100) and divide it into two pieces: the first 14 bits are the page, and the last 10 are the offset. Replace the first 14 bits (00000000000001) by (00000000010), to get the physical address 000000000100111110100, or 2548.

e. Physical address 1023 is at offset 1023 in frame 0. Virtual page 3 maps to frame 0, so this is virtual address 4095.

(PHYSICAL ADDRESS) 000000000001111111111 → 000000000000111111111111 (virtual)

2. You have a virtual memory system with a two-entry TLB, a 2-way set associative cache and a page table for a process ***X***. Assume cache blocks of 8 words and page size of 16 words.
In the system below, main memory is divided up into blocks, where each block is represented by a letter. Two blocks equals one frame.



TLB

Page	Frame
0	3
4	1

Cache

	TAG	DATA	TAG	DATA
Set 0	00	C	01	I
Set 1	00	D	10	H

Page Table

	Frame	Valid
0	3	1
1	0	1
2	-	0
3	2	1
4	1	1
5	-	0
6	-	0
7	-	0

Main Memory

Frame		Block
0	C	0
	D	1
1	I	2
	J	3
2	G	4
	H	5
3	A	6
	B	7

Virtual Memory For Process X

Page		Block
0	A	0
	B	1
1	C	2
	D	3
2	E	4
	F	5
3	G	6
	H	7
4	I	8
	J	9
5	K	10
	L	11
6	M	12
	N	13
7	O	14
	P	15

Given the system state as depicted above, answer the following questions:
a. How many bits are in a virtual address for process ***X***? Explain.
b. How many bits are in a physical address? Explain.
c. Show the address format for virtual address 18_{10} (specify field name and size) that would be used by the system to translate to a physical address and then translate this virtual address into the corresponding physical address. (Hint: convert 18 to its binary equivalent and divide it into the appropriate fields.) Explain how these fields are used to translate to the corresponding physical address.
d. Given virtual address 6_{10} converts to physical address 54_{10}. Show the format for a physical address (specify the field names and sizes) that is used to determine the cache location for this address. Explain how to use this format to determine where physical address 54 would be located in cache. (Hint: convert 54 to binary and divide it into the appropriate fields.)
e. Given virtual address 25_{10} is located on virtual page 1, offset 9. Indicate exactly how this address would be translated to its corresponding physical address and how the data would be accessed. Include in your explanation how the TLB, page table, cache and memory are used.

Ans.
a. $2^3 * 2^4 = 2^7$, so there are 7 bits in a virtual address

b. $2^2 * 2^4 = 2^6$, so there are 6 bits in a virtual address

c. 18 = 001 0010 (where 001 is the page field and 0010 is the offset). Using the page table, and going to entry 1, we see that page 1 maps to frame 0, so the actual physical address would be 00 0010, or 2.

d. 54 = 11 0 110, where 11 is the tag, 0 is the set, and 110 is the offset. Therefore, this would map to Set 0 in cache. Once there, if the tag is found, the block is in cache. If not, it is a miss.
e. 25 = 001 1001, where 001 is the virtual page, and 1001 is the offset. (This maps to physical page 00 with offset 1001.) The TLB would first be checked to see if the pair (1,0) was present (virtual page 1, physical frame 0). If so, 00 can be substituted for 001, giving the actual physical address 00 1001. If the entry is not found in the TLB, the page table must be accessed, at which time the physical address 00 1001 is determined. At this point (regardless of whether we found the physical address using the TLB or the page table), the cache is checked to see if the block containing this physical address is currently cached. The memory address 001001 is divided up into 00 1 001, where 00 is the tag, 1 is the set, and 001 is the offset. Set 1 in cache is then checked for the tag 00 (it would be found as address 25 is in block D). The value in cache is used. (If it had not been found in cache, main memory would be accessed.)