CSE321 - Operating Systems

A process runs in a system with multi level paging and it has a logical address space of 18 bits. In the system page size is 128 Bytes, size of each entry of the page table is 8 Bytes and size of the main memory is 512 KB. In order to fit the pages of the process in the main memory the OS applies a two-level paging technique in outer page number bits of the logical address space until the outer most page table can be allocated in a frame of the main memory.

a. Illustrate the logical address space of the process including the necessary outer page bits, inner page bits and offset bits of every step with proper mathematical calculations during the paging mechanism of the system described above.
b. In this system, if the CPU generates logical addresses 80211 and 153114 then map the corresponding physical addresses of these logical addresses.

Necessary page table information is given below

2nd Outer table	
Page #	Frame #
0	8
1	2
2	6
3	
4	1
5	
6	
7	

Outer Table			
Frame 1		Frame 6	
Page #	Frame #	Page #	Frame #
0	295	0	854
1		1	
2		2	165
3		3	
4	30	4	
5		5	
6		6	
7	628	7	36
8		8	61
9	49	9	
10	510	10	
11		11	
12		12	568
13	351	13	
14		14	
15		15	

Inner Table			
Frame 36		Frame 510	
Page #	Frame #	Page #	Frame #
0	498	0	
1		1	
2	489	2	
3	65	3	
4		4	
5		5	
6		6	
7	687	7	948
8		8	
9		9	1051
10		10	
11	563	11	
12		12	716
13		13	497
14		14	
15		15	

Answer: i. The goal of this problem is to allocate page tables such that each can fit One frame size. Here, we first do some pre-calculations:

Logical Address **space** (not size) = 18 **bits**

Page **size** (which is also frame size) = 128 **bytes** = 2^7 bytes
(Since the index of 2 is 7, offset is of 7 bits)

Page Table Entry (PTE) **size** = 8 **bytes** = 2^3 bytes

Here, you can consider each row as a page table entry/ PTE, and the size occupied by each row as PTE size.
For example, in the given 2nd outer page table, there are 8 entries. If I multiply 8 by PTE size, i.e, 8 entries * 8 bytes PTE size = 64 bytes Page table size < 128 bytes Frame size. So, the 2nd outer page table can fit in one frame.
Again, consider the case of both the 1st outer page table and Inner page table which contains 16 entries each. In this case, 16 entries * 8 bytes PTE size = 128 bytes Page table size = 128 bytes Frame size. So, these two tables are exactly the same size as frame size.

Moving on to calculations,
We first leave off the last 7 bits as offset bits from the 18 bit logical address space as calculated above. Remember, in this case we are dividing the logical address space **<u>from right to left.</u>**

18 bit address space

11 bit for paging	7 bit offset

Here, the following process will be done in a loop until the page table size <= frame size:

No. of page table entries = 2^(leftmost bits)
= $\mathbf{2^{11}}$
We multiply the obtained no. of entries with PTE size to find if the page table fits the frame size

$2^{11} * 2^3 = 2^{14} > 2^7$ frame size.

Since the page table size is larger than frame size, we need to do two-level paging here. From the left side of offset, we set the following number of bits as

inner table bits = Offset bit - **PTE size index** = 7 - 3 = 4 bits

(Why this calculation? Because each table should have a size of maximum 2^7 bytes. If size of each row/ page table entry is 2^3 bytes, then maximum number of rows / entries = $2^7 / 2^3 = 2^{7-3} = 2^4$. So basically, 4 bits will be used in address translation for the page table)

Point to remember that only the page table on the left side of offset is called inner table
The address space then becomes:
18 bit address space

7 bits	Inner page table 4 bits	7 bit offset

—--
Again, we follow the loop:

(Here, we are basically considering inner page table bit as new offset and the remaining bits to the left for paging)
No. of new page table entries = 2 ^ (leftmost bits)
= $\mathbf{2^7}$

We multiply the obtained no. of entries with PTE size to find if the page table fits the frame size

$$2^7 * 2^3 = 2^{10} > 2^7 \text{ frame size.}$$

Since the page table size is larger than frame size, we need to do two-level paging here. From the left side of the inner table, we will call it the

outer table bits = Offset bit - **PTE size index** = 7 - 3 = 4 bits

Point to remember that only the page table on the left side of offset is called inner table
The address space then becomes:

18 bit address space

3 bits	Outer page table 4 bits	Inner page table 4 bits	7 bit offset

—---

Now, we don't need to follow the loop any longer as we can see that the remaining 3 bits will obviously be enough to fit in a frame. However, to show the outcome in a calculation:

No. of new page table entries = 2 ^ (leftmost bits)
$= 2^3$

We multiply the obtained no. of entries with PTE size to find if the page table fits the frame size

$$2^3 * 2^3 = 2^6 < 2^7 \text{ frame size.}$$

Since the newest page table fits within the frame size, the following table shows the resulting hierarchical paging structure of the address space:

2nd outer page table 3 bits	Outer page table 4 bits	Inner page table 4 bits	7 bit offset

18 bit address space

—---

Answer ii. Using the obtained hierarchical paging structure, we find the physical address from the logical address. In case of logical address 80211:
Binary form: 010 0111 0010 1010011 (broken down according to the hierarchical structure)

Here, we are going to traverse frames from **left to right** as given in the question.

For 010 which is in 2nd outer page table: $(010)_2 \rightarrow (2)_{10}$ (page#) → 6 (frame #)

In the Frame 6 obtained from above in outer page table:

$(0111)_2 \rightarrow (7)_{10}$ (page #) → 36 (frame #)

In the Frame 36 obtained from above in inner page table:

$(0010)_2 \rightarrow (2)_{10}$ (page #) $\rightarrow$ 489 (frame #)

This final obtained frame number will be used to find the physical address.

$(489)_{10} \rightarrow (000111101001)_2$
Appending offset bits:
$(000111101001\ 1010011)_2 \rightarrow (62675)_{10}$ $\rightarrow$ Required physical address in decimal number system