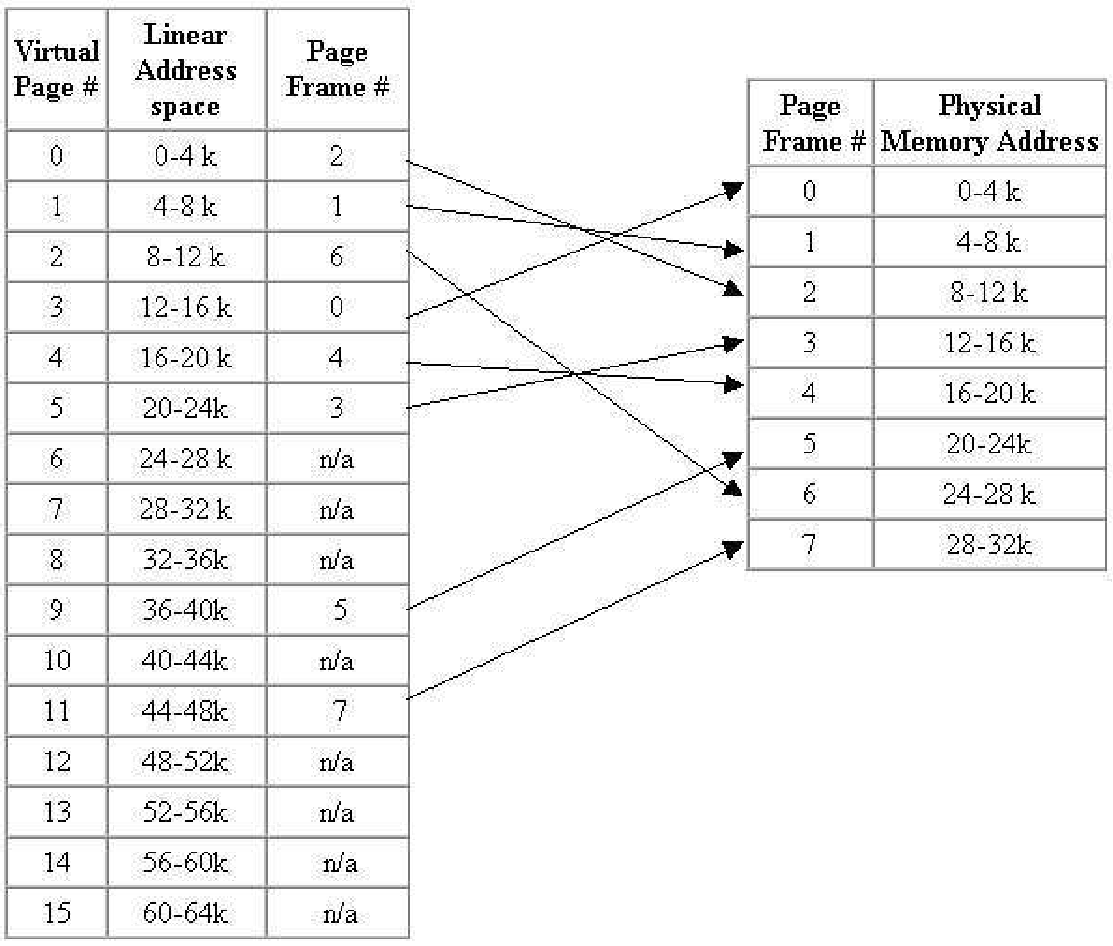
Assignment - Linear-to-Physical Address Translation

Memory snapshot:



Virtual Page #	Linear Address space	Page Frame #
0	0-4 k	2
1	4-8 k	1
2	8-12 k	6
3	12-16 k	0
4	16-20 k	4
5	20-24k	3
6	24-28 k	n/a
7	28-32 k	n/a
8	32-36k	n/a
9	36-40k	5
10	40-44k	n/a
11	44-48k	7
12	48-52k	n/a
13	52-56k	n/a
14	56-60k	n/a
15	60-64k	n/a

Page Frame #	Physical Memory Address
0	0-4 k
1	4-8 k
2	8-12 k
3	12-16 k
4	16-20 k
5	20-24k
6	24-28 k
7	28-32k

The "paging unit" within the CPU takes "linear" addresses and is responsible for translating them into "physical" ones. This process must account for the fact that the linearly-addressed virtual pages are being continually shuffled around among the physical page frames. Two identical requests to read a certain linear address with constant content, in order to deliver that same content, may require reading from one physical part of memory on the first occasion and a different one on the next. Programs supply the paging unit with a linear address and in effect ask it, "Give me the data there." And every time the paging unit must in effect ask itself "Where is 'there?'" It fetches the info only after answering the question to its own satisfaction (including some writing between disk and memory if needed).

The above diagram shows a snapshot of memory configuration at a given moment. At that moment if software requested access to byte 0, the physical memory fetch would be from cell 8192 after the paging unit did its work. The paging unit, intermediating between the requesting software and the physical memory, makes that numerical transformation. You can make it yourself by observing the arrow that maps the page containing byte 0 of linear memory (page #0) into the 3rd frame (frame #2).

That frame consists of physical memory's bytes 8192 through 12287. The byte in question is the first in its range: 0 on the virtual page and so 8192 in the physical frame.

Let's do a couple of examples. What about linear memory's bytes 10000, 20000, and 30000? Into which page does address 10000 fall? Examination of the scale below shows it's page #2, since 10000 falls within 8192-12287. Examination of the figure above reveals the page frame where page #2's content currently resides; it's frame #6 (follow the arrow from virtual page #2). That is, frame 6 contains page 2. Observe that byte 10000 is 1808 bytes deep into page 2 (10000 minus 8192). So the memory cell containing 10000's content must be at the same offset within the containing frame. The starting address of the frame being 24576, the byte we are looking for must be 26384 (24576 plus 1808). For linear address 20000, virtual page 4 contains it at an offset of 3616 (20000-16384). And virtual page 4 maps to physical frame 4. By coincidence then, linear address 20000 maps to physical address 20000. And for 30000, we can't do the problem. The page it belongs to (page 7) is absent from physical memory. We can't give the physical address of the memory cell that holds linear address 30000's content if there is no such cell. (The paging system would do something to remedy that in real life, putting that content from disk into one of the frames for us; that's the whole purpose of the system. But given the static snapshot we are dealing with the question as asked is answerless.)

This table gives exact address boundaries for pages, in decimal and equivalently in hexadecimal:

Page or page frame #	first/last address-dec / first/last address-hex
0	0 - 4095
	0h - FFFh
1	4096 - 8191
	1000h - 1FFFh
2	8192 - 12287
	2000h - 2FFFh
3	12288 - 16383
	3000h -3FFFh
4	16384 - 20479
	4000h - 4FFFh
5	20480 - 24575
	5000h - 5FFFh
6	24576 - 28671
	6000h - 6FFFh
7	28672 - 32767
	7000h - 7FFFh
8	32768 - 36863
	8000h - 8FFFh
9	36864 - 40959
	9000h - 9FFFh
10	40960 - 45055

	A000h - AFFFh
11	45056 - 49151
	B000h - BFFFh
12	49152 - 53247
	C000h - CFFFh
13	53248 - 57343
	D000h - DFFF
14	57344 - 61439
	E000h - EFFFh
15	61440 - 65535
	F000h - FFFFh

Assignment:

Below are 5 linear addresses to be converted to physical ones, and 5 physical addresses to be converted to linear ones. Please submit answers to them onto the remote Unix machine using these preparation and submittal instructions. The instructions assume multiple-choice questions, but these are not multiple choice questions. Simply write their non-multiple-choice answer on the line you type for each question (remembering to number the lines with the question numbers). If a question is unanswerable for some reason, write "n/a" as its answer. Otherwise write a single number as the answer. All numbers below are in the decimal numbering system. Please write your answers in decimal also.

The following are linear memory addresses. For each, given the above snapshot, what is the physical memory address that holds its content? If an address resides in a page not currently mapped to physical memory, so indicate by writing "n/a".

1. 12287
2. 12288
3. 25000
4. 39151
5. 46000

The following are physical memory addresses. For each, what is the linear memory address whose content it holds?

6. 0
7. 28671
8. 28672
9. 30000
10. 64535