How Large is the TLB?

Segmentation vs. Paging

- **Segmentation**: divide memory into variable-size pieces
- Problems with external fragmentation

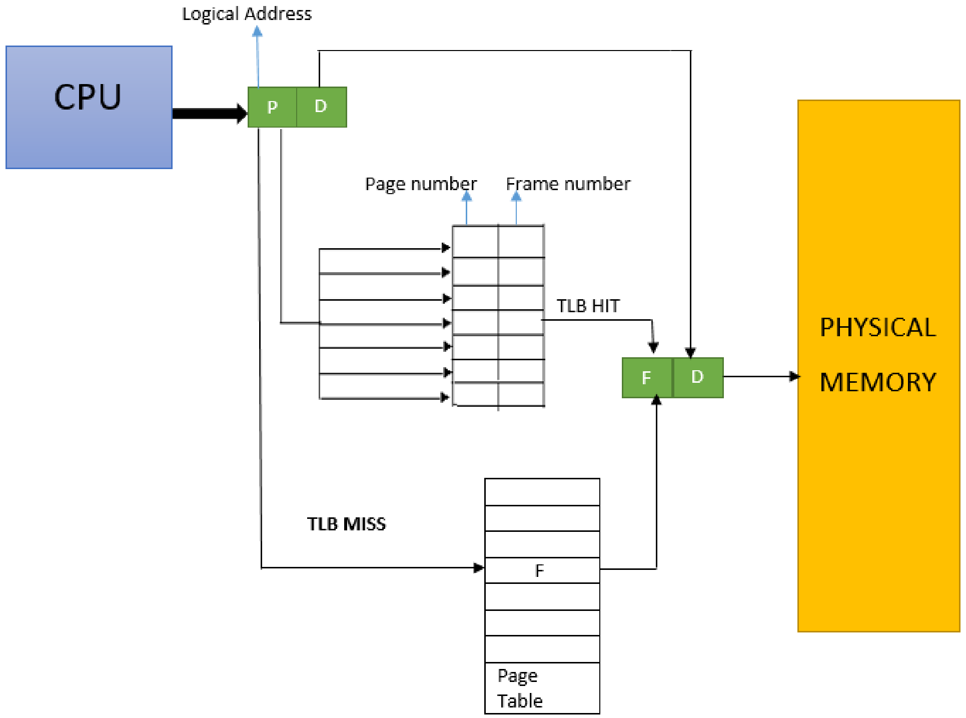
- **Paging**: divide memory into fixed-size pieces (pages)
- Problem with slow address translation, page table stored in physical memory
- Solution: hardware support, the Translation Lookaside Buffer (TLB)

What is the TLB?

- A hardware cache for page table entries
- Part of MMU
- Stores translations from virtual addresses to physical addresses
- Makes paging possible

TLB Hit

- 1 clock cycle
- No physical memory lookup



TLB Miss

- 10 - 100 clock cycles
- Requires lookup in the page table

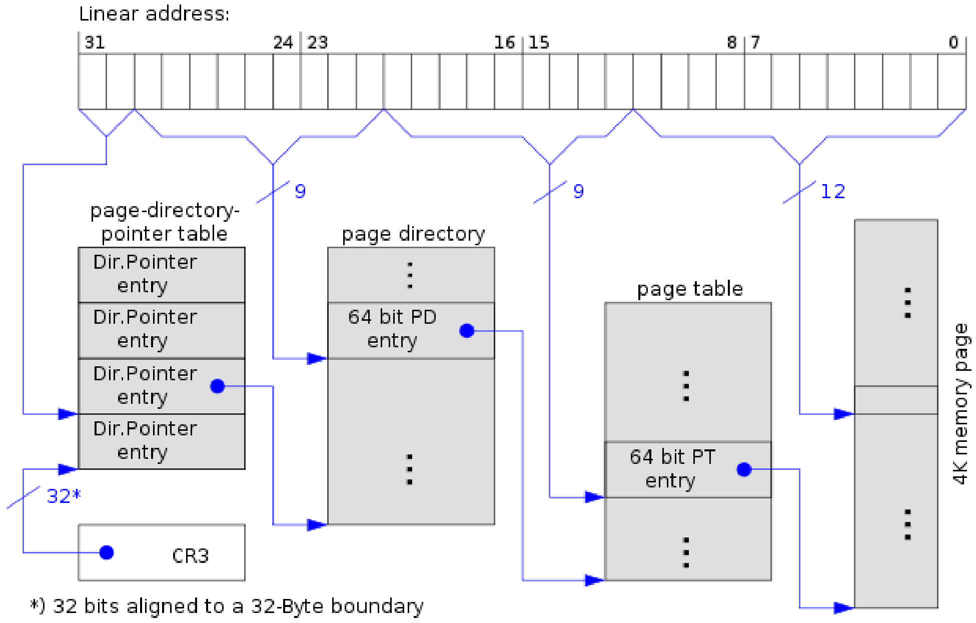


Image by RokerHRO - eigene Arbeit / own work using Xfig., CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=3741876

Regular Cache Effects or the TLB?

Regular CPU Cache (L1 - L3)

- Caches contents of the memory

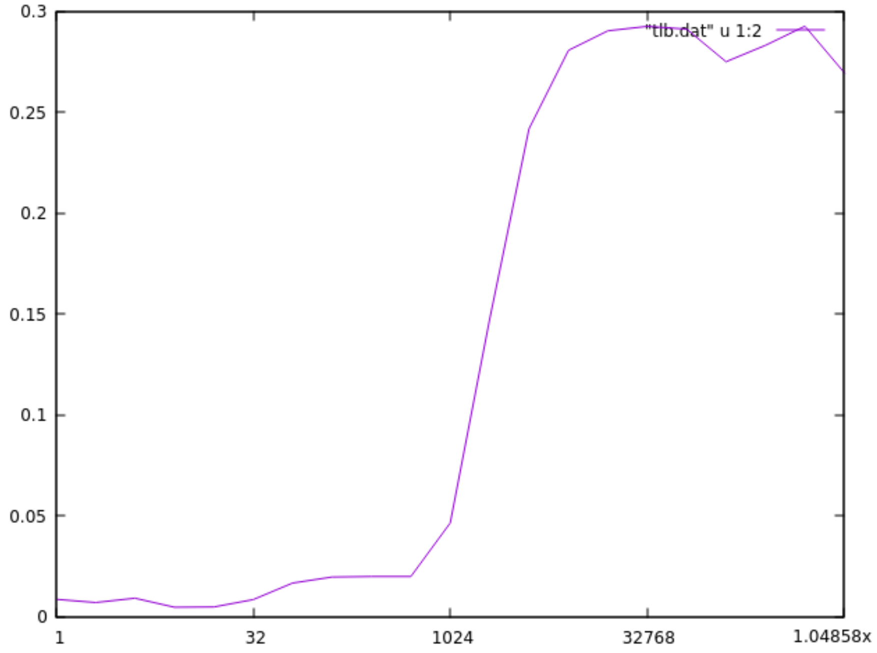
TLB

- Caches the page table
- Makes sure finding the memory is quick

Both

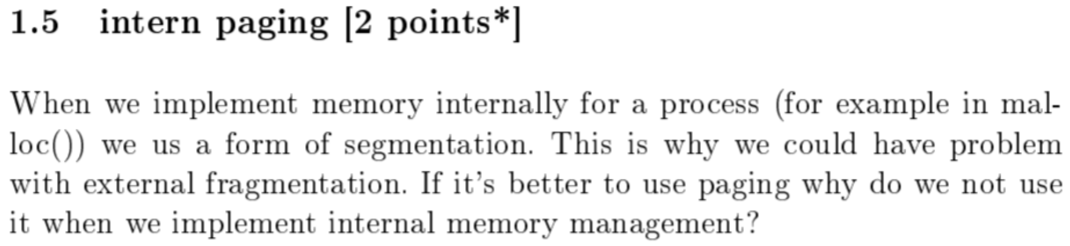
- Sequential access is the way to go!

Result



Exam Questions

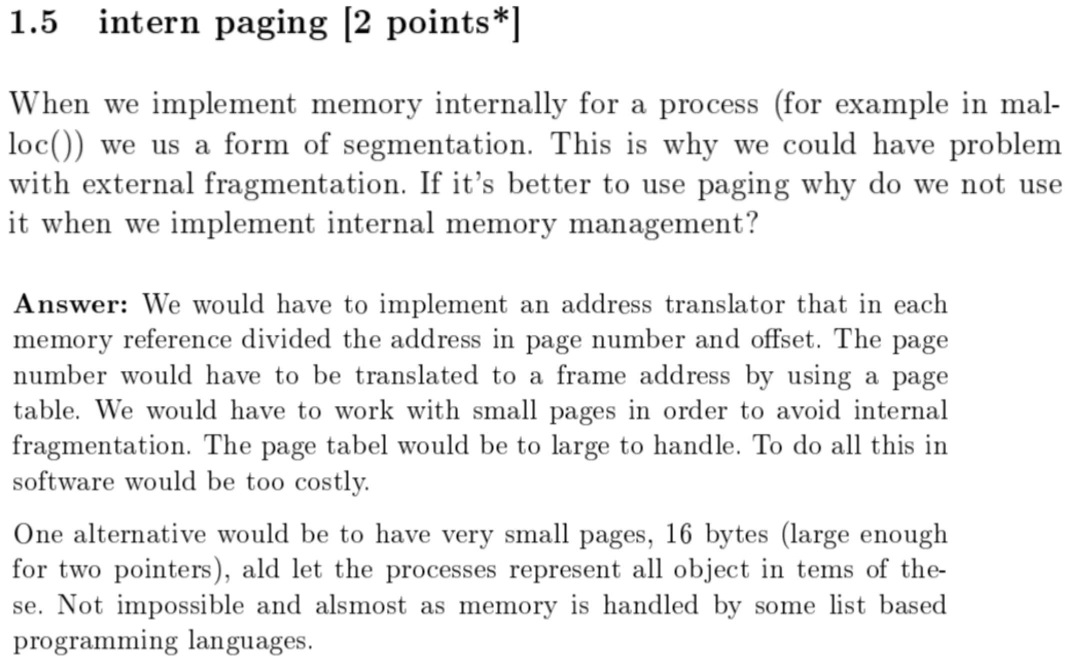
Exam Question 1



1.5 intern paging [2 points*]

When we implement memory internally for a process (for example in malloc()) we us a form of segmentation. This is why we could have problem with external fragmentation. If it's better to use paging why do we not use it when we implement internal memory management?

Exam Question 1



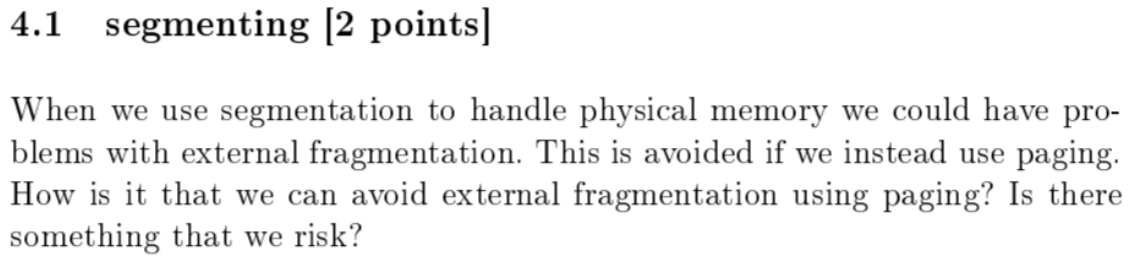
1.5 intern paging [2 points*]

When we implement memory internally for a process (for example in malloc()) we us a form of segmentation. This is why we could have problem with external fragmentation. If it's better to use paging why do we not use it when we implement internal memory management?

Answer: We would have to implement an address translator that in each memory reference divided the address in page number and offset. The page number would have to be translated to a frame address by using a page table. We would have to work with small pages in order to avoid internal fragmentation. The page tabel would be to large to handle. To do all this in software would be too costly.

One alternative would be to have very small pages, 16 bytes (large enough for two pointers), ald let the processes represent all object in tems of these. Not impossible and alsmost as memory is handled by some list based programming languages.

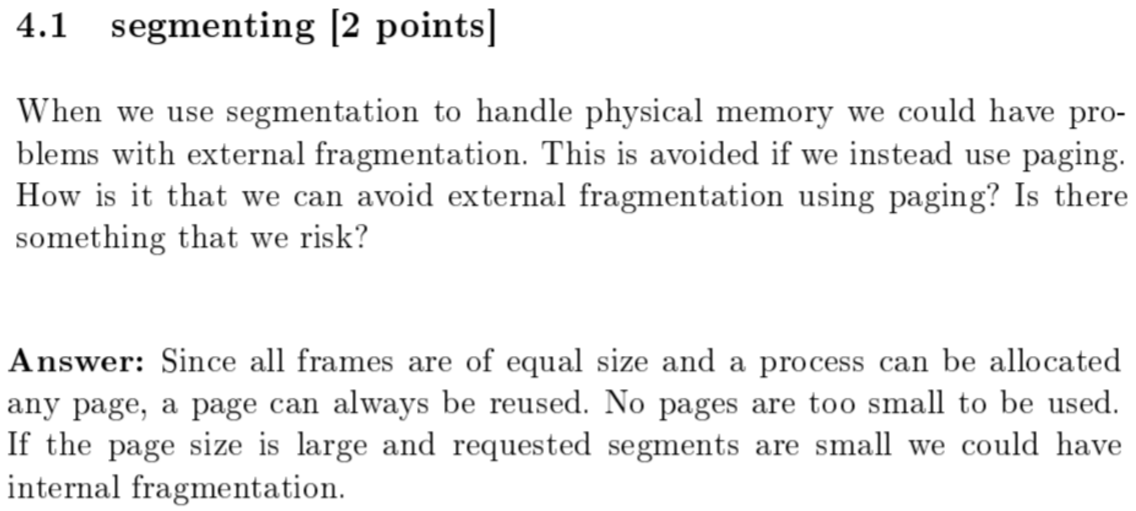
Exam Question 2



4.1 segmenting [2 points]

When we use segmentation to handle physical memory we could have problems with external fragmentation. This is avoided if we instead use paging. How is it that we can avoid external fragmentation using paging? Is there something that we risk?

Exam Question 2

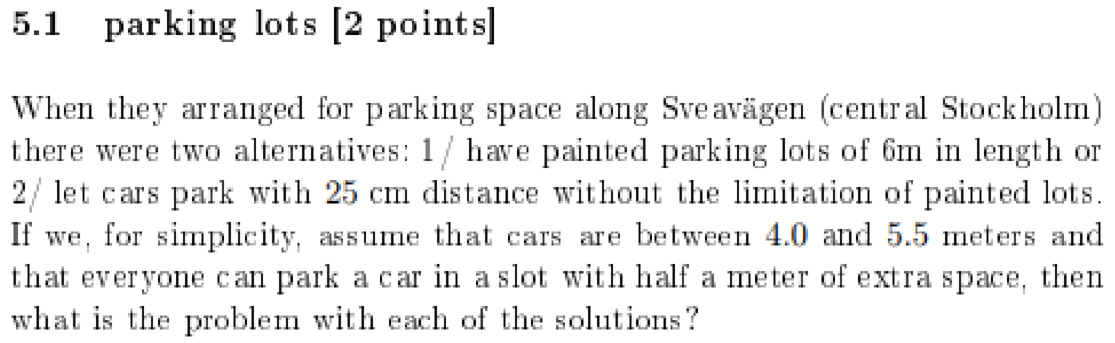


4.1 segmenting [2 points]

When we use segmentation to handle physical memory we could have problems with external fragmentation. This is avoided if we instead use paging. How is it that we can avoid external fragmentation using paging? Is there something that we risk?

Answer: Since all frames are of equal size and a process can be allocated any page, a page can always be reused. No pages are too small to be used. If the page size is large and requested segments are small we could have internal fragmentation.

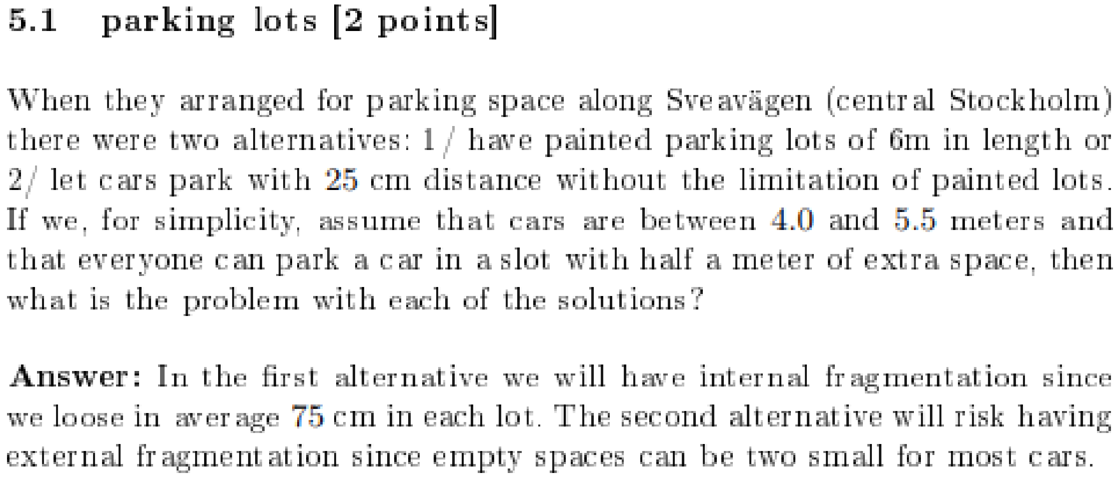
Exam Question 3



5.1 parking lots [2 points]

When they arranged for parking space along Sveavägen (central Stockholm) there were two alternatives: 1/ have painted parking lots of 6m in length or 2/ let cars park with 25 cm distance without the limitation of painted lots. If we, for simplicity, assume that cars are between 4.0 and 5.5 meters and that everyone can park a car in a slot with half a meter of extra space, then what is the problem with each of the solutions?

Exam Question 3

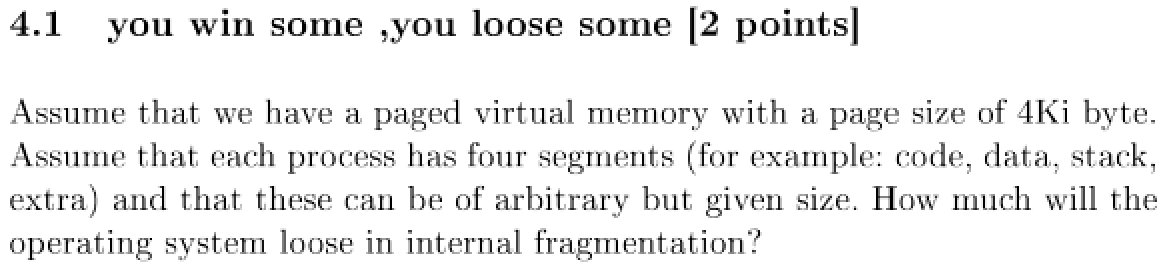


5.1 parking lots [2 points]

When they arranged for parking space along Sveavägen (central Stockholm) there were two alternatives: 1/ have painted parking lots of 6m in length or 2/ let cars park with 25 cm distance without the limitation of painted lots. If we, for simplicity, assume that cars are between 4.0 and 5.5 meters and that everyone can park a car in a slot with half a meter of extra space, then what is the problem with each of the solutions?

Answer: In the first alternative we will have internal fragmentation since we loose in average 75 cm in each lot. The second alternative will risk having external fragmentation since empty spaces can be two small for most cars.

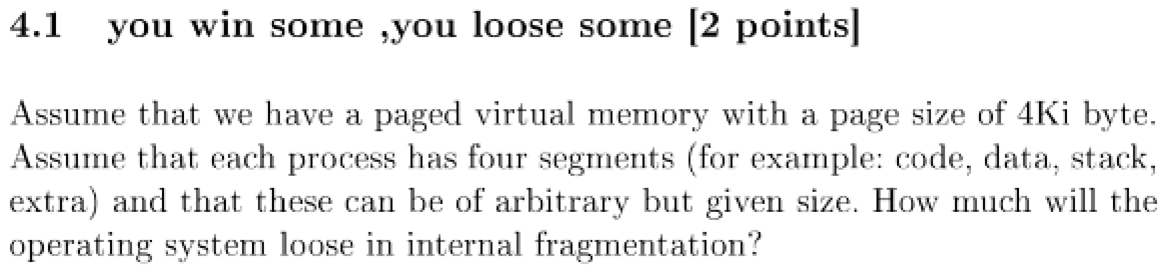
Exam question 4



4.1 you win some ,you loose some [2 points]

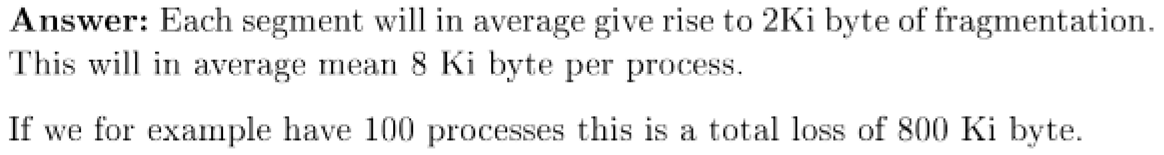
Assume that we have a paged virtual memory with a page size of 4Ki byte. Assume that each process has four segments (for example: code, data, stack, extra) and that these can be of arbitrary but given size. How much will the operating system loose in internal fragmentation?

Exam question 4



4.1 you win some ,you loose some [2 points]

Assume that we have a paged virtual memory with a page size of 4Ki byte. Assume that each process has four segments (for example: code, data, stack, extra) and that these can be of arbitrary but given size. How much will the operating system loose in internal fragmentation?



Answer: Each segment will in average give rise to 2Ki byte of fragmentation. This will in average mean 8 Ki byte per process.

If we for example have 100 processes this is a total loss of 800 Ki byte.