Operating Systems

Opdracht 6

8.17

a. 100 nanoseconds, 50 nanoseconds to access the frame in the page table and 50 nanoseconds to access the word with the offset in memory.
b. We need 100 nanoseconds to access a word not stored in the TLB and 52 nanoseconds for words already stored in the TLB, this way we can calculate the average time it takes to access a word in memory: 0.75 * 52 + 0.25 * 100 = 64 nanoseconds on average.

8.21

When a system has a lot of physical memory adresses, the page table can become very large. When paging the page table, the pages currently not used can be forgotten and not stored in memory. This way the page table does not have to be stored in memory completely.

9.1

- TLB miss with no page fault: The page is in memory but not in the TLB.
- TLB miss and page fault: The page is not in the TLB and not yet loaded into memory.
- TLB hit and no page fault: The page is in the TLB as it is in memory.
- TLB hit and page fault: This can't happen because for a TLB hit to happen, the memory block has to be in the TLB and for it to be in the TLB, the memory block has to first be in memory.

9.7

No, because there are multiple kernel threads, the other threads can continue running while the blocked one has to wait for the page to be loaded into memory.

9.18

80 percent of the time, the access time is equal to one memory reference:

0.8 * 1 = 0.8 microseconds

2 percent of the time, a page fault occurs:

0.02 * 2020 = 40.4 microseconds

18 percent of the time, a normal memory reference occurs:

0.18 * 2 = 0.36 microseconds

All added together, this gives us an avarage time of:

0.8 + 40.4 + 0.36 = 41.56 microseconds

9.25

The advantages of such a paging scheme is that the size of a page table could be decreased because large pieces of memory would only need one page. For this to work, the system would have to maintain the size of each page in the paging table and, when calculating the real physical adress, would have to take the page size of a specific page into account.