Operating Systems

Opdracht 5

8.1

External fragmentation arises if many processes are loaded onto memory and are then deallocated. This type of fragmentation is based on the memory that is lost between the memory each process occupies. Internal fragmentation however arises when memory is split into blocks and a process does not fit in a certain amount of blocks. When the block size is, for example, 16MB and the process size is 30MB. The process will get two blocks of memory and wasts 2MBs.

8.5

Memory allocation:

a) External fragmentation does not exist when using memory allocation because the memory is splitted into partitions and no memory between the partitions is lost.
b) Internal fragmentation does exist when using memory allocation because when a process is much smaller than the partition size, a lot of memory in that partition will be unused and thus be lost.
c) Code sharing is not possible with memory allocation because each process has an independent piece of memory allocated.

Segmentation:

a) External fragmentation does exists when using pure segmentation because when a large process finishes executing and that memory is freed, then if a smaller process takes that place, it will leave an even smaller portion of memory fragmented.
b) Internal fragmentation does not exist when using segmentation because each process gets exactly the amount of memory it needs.
c) Code sharing is not possible when using segmentation because each process has an independed piece of memory allocated.

Paging:

a) Paging does fix external fragmentation because the memory is split into blocks. A process then gets blocks assigned.
b) Paging does not fix internal fragmentation because the blocks of memory will (almost every time) not be filled completely.
c) Code sharing is possible when using paging because we can virtual adress spaces to the same physical adress space.

8.7

The space constraint is a reason why mobile operating systems do not support swapping. Also, most mobile systems use flash memory that can only be written to a fixed amount of times before the memory fails.

8.13

a) With a single level page table, the physical adresses are the same length as the virtual adresses, this means that the total amount of entries equals: $2^{21}/2^{11} = 2^{10}$.
b) With an inverted page table, the virtual adress also stores the process ID. This means we now have only: $2^{16}/2^{11} = 2^{5}$ entries.