Report on exercise #2

Matteo Corain S256654

Laboratory #4 – System and device programming – A.Y. 2018-19

The proposed solution to find out the number of pages that the minimal kernel allocated makes essentially use of a for loop that, for each iteration, tries to access a page until a page fault and a consequent kernel panic are produced. After having initialized the segmentation and the paging tables by means of the init_descriptor_tables() and init_paging() functions, the loop is set up by setting the value of variables i (u32int) which is used to count the index of the page that is being checked, and ptr (u32int*), which is used as the pointer to reference the different pages, to 0. At each iteration, the loop:

- Accesses the memory location addressed by ptr, storing the result in the variable data (u32int);
- If no page fault has occurred (i.e. no kernel panic has been generated), writes a message on the screen by means of the monitor_write() function and its derivatives for integer numbers.

After each iteration, the value of i is incremented and the value of PAGE_SIZ divided by 4 (i.e. shifted by 2) is added to ptr. PAGE_SIZ is defined as a constant set to 4096 (0x1000 in hexadecimal notation), which represents the size of each page in the Intel x86 architecture (4 KB); it is divided by 4 since, due to the way pointers are handled in the C language, each increment of a pointer referring to a 32-bits long type will effectively result in a memory access shifted by 32 bits with respect to the pre-increment value of the pointer.

As a result of the execution of the program, we obtain that the minimal kernel allocates exactly 264 pages: the first page fault, in fact, is generated when accessing address 0x10900, corresponding to the beginning of the 265th page.