The bootloader loads in 16bit real mode and goes to 32 bit protected mode, for which it uses the Global Descriptor Table to allocate memory to the process which would be running once we switch to protected mode this table of descriptors is useful to keep a track of processes trying to write in uninitialised and addresses that are out of bounds.

Inside the program I have use the right operation to shift the content of the cr0 register to the right and then add with carry a 1 to the content of eax register after that I shift the value inside eax to the address to which ebx points (which is the address of VGA text buffer).

For outputting the string I have used the lodsb command which loads each bytes into al and then displays on the screen. If at any point of time al is 0 then I move to the next segment.