

Higher Half Setup

After enabling 64bit mode and properly initialising system interrupts, we made the kernel run in the higher half of memory.

Linker Script Trick

Using a [well-known trick](#), we made the 64bit kernel use a virtual address as its starting address as shown in the linker file located at the root of the COS repository.

To do so we define a virtual address located into “High Memory” to force the CPU to load the code at this address and thus generate all kernel addresses as virtual addresses starting with the prefix we chose (in this case, addresses starting with 0xFFFFFFF8).

Therefore, the kernel while the kernel will be physically loaded at address 0x10000 as defined by the linker script, the CPU will interpret all kernel address references as “High Memory” addresses.

Paging Trick

In the file **./src/boot/main.asm**, a new entry was added as the last entry of the Page Map Level 4 to handle the new load address of the kernel.

Due to the way Paging Level 4 works, a virtual address contains only 48bits interpreted by the CPU.

Bit 48 through 39 define the entry used by the Page Map Level 4.

As 0xFFFFFFF80000000000 is a 64bit value, the only significant value is 0xFF8000000000. The upper 9bits are 0b11111111 which is 511.

Since each paging level only has 512 entries, this is the reason why it was added last.

As identity mapping is still needed during initialisation because the code is still executed with physical addresses in “Low memory”, the first entry is kept.

Higher Half Initialisation

The higher half initialisation was added in the file **./src/boot/main64.asm**.

As the current code execution still uses physical addressing, near jumping to a label would use physical addressing instead of virtual addressing.

The mov instruction is used to load the rax register with the 64bit value representing the label and an indirect jump is made to it. This allows the CPU to switch execution to virtual addresses mapped earlier. Once done, the stack is reloaded with the virtual address.

Since only user space program will be loaded with low addresses, we undo the identity map by removing the first entry of PML4 and we reload the table's address into Control Register 3.

The Translation Lookaside Buffer is a hardware component used for storing virtual addresses translations that are most used during the current cycles of execution.

It is used by the CPU for performance as it won't need to decode virtual addresses via its MMU.

Since the execution has moved into "Higher Half", some translations included in the TLB might include unmapped virtual addresses used in the identity map.

While it would therefore be required to be flushed with the INVPLG instruction, reloading the table into Control Register 3 will force this TLB flush instead.

As the old mapping was removed, the GDT must also be updated because the CPU still uses its identity mapped address.

In 64bit mode, far jumps don't exist.

After jumping to higher half, the segment registers must be reloaded with the new base addresses.

As CS can't be updated directly, a far return is used instead of a far jump.

The RETFQ instruction will take two values from the stack and update the CS and RID registers accordingly.