# No code submission required

# Read the project description (https://fxlin.github.io/p1-kernel/) before proceeding

# Part A: Q&A

This assignment refers to the code under **exp6/**.

1. (5) How many levels of pgtables will we have, if we map memory in sections (2MB)?

There will be three levels of page tables if we map in sections (2MB).

1. (5) What’s the base address of kernel’s virtual address space? What’s the rationale for choosing such a value?

The base address of the kernel’s virtual address space is 0x106000.

1. (5) What’s the base address of each process’s virtual address space? The rationale for choosing such a value?
2. (5) When kernel just starts and MMU is off, the CPU should be accessing memory using physical addresses. If that’s the case, how could the kernel possibly access variables (e.g. bss\_start) or functions (e.g. memzero), which are linked at virtual addresses (see pictures below)?

Text

Description automatically generated

(Hint: check the disassembly of the kernel binary that access the variables/functions)

1. (2)

Right after kernel switches to EL1 and clears the BSS, the kernel populates its pgtables via \_\_create\_page\_tables:

// boot.S

\_\_create\_page\_tables:

mov x29, x30 // save return address

The first thing \_\_create\_page\_tables does is to save LR (i.e. x30) to x29. LR points to the address that \_\_create\_page\_tables will return to. Normally, a function would save LR on its stack; this avoids losing the LR value when this function invokes another function with instruction BL, which overwrites LR; when such a function returns, it pops the saved LR value from its stack.

Is it a bug that we save LR to x29?

Instead of saving LR to x29, can we save LR by pushing it to stack? Try it out yourself and explain your observation.

*Changelog*

*Jan 2024. Clarification.*