# Convert this docx to PDF before submitting it

# Attach a screenshot of you successfully running the given code

i.e. a screenshot of your emulator/rpi3 terminal printing "hello world".

# First, inspect the kernel binary (kernel8.elf) of p1exp1.

You may use command line tools (e.g. objdump or readelf) or GUI tools (e.g. ODA). Btw, we have provided instructions <https://fxlin.github.io/p1-kernel/dump/>

**How many sections are in the elf file? What are these sections?**

Include your command line output as text (if you use command line tools) or attach screenshots (if you use GUI tools).

**How many symbols are in the elf file?**

**What is the address of symbol kernel\_main? What are the first 8 bytes at the symbol? What are the corresponding instructions?**

**How many bytes does each aarch64 instruction contain?**

**Now examine kernel8.img (use the hexdump command or the** [**VSCode plugin**](https://marketplace.visualstudio.com/items?itemName=slevesque.vscode-hexdump)**). Search for the first 8 bytes of kernel\_main(). Can you find it? At which offset of kernel8.img?**

**How is kernel8.img generated out of kernel8.elf?**

# Second, some ARM64 exercise:

**How many general-purpose registers in aarch64 (i.e. the 64-bit execution state of ARM64)? How many bytes in each register?**

There are 31 general purpose registers, x0 – x30 where each register are of size 64-bit or 8 bytes.

**Use your own words, explain the following instructions, each in one short sentence.**

**and**: performs the logical AND operation between two values

**bl**: does a unconditional branch to a target address and stores the return address to the general purpose register x30; the ret instruction can be used to branch back to the target address

**mov**: moves a constant or a register value to another register

**adr**: grabs the relative address of a label and loads it into a target register

**mrs**: takes a value from the system register and loads it into one of the general purpose registers, that is x0 – x30

**Back to kernel8.elf. Use your own words, explain instruction by instruction: how the delay() function works.**



# Third, about Rpi3

Watch the video [Eben Upton on Rpi3](https://youtu.be/37_7arZZlUI) and answer the questions:

What would be the benefit of supporting 64-bit (AArch64)? In particular, why it is important to support 64-bit by an operating system?

Have you ever used Raspberry in any chance? What was for?