Exp 1

- linker script: how object files (.o) map to output file (.elf)
  - .text.boot: stores kernel startup code
  - .text: kernel instructions
  - .rodata: read-only data
  - .data: stores global data
  - .bss: contains data that need to be initialized to 0

- ARM64
  - x0 - x30: general registers; 64 bit
  - mrs: load value from system register to general purpose register
  - and: performs AND operation
  - cbz: compare prev. operation to 0 + jump
  - b: unconditional jump
  - adr: load label's address to target register
  - sub: subtract values
  - bl: branch w/ link
  - mov: move value from register or a constant to target register

- access to devices performed via memory-mapped registers
- UART: simple software device allowing software to send out text characters to a diff. machine
- GPIO: general purpose input/output
  - ↳ provides registers
  - ↳ each bit corresponds to a pin on Rpi3
- pull-up: if nothing is connected; return 1 from pin
- pull-down: " " return 0 from pin

- pin state preserved even after boot

- `addr2line -e <elf file> <address>` } lookup source line
    - -e specifies ELF file

- list all symbols & addresses
  ```
  nm build/kernel8.elf
  ```
    - format: "link address" - "symbol table" - "symbol name"

- dump section as raw data
  ```
  aarch64-linux-gnu-objdump -s -j <section> <.elf file | others>
  ```

- disassemble specific address range
  ```
  aarch64-linux-gnu-objdump -dS <.elf file | others>
  --start-address=<start address> --stop-address=<stop address>
  ```

- disassemble a specific function
  ```
  gdb-multiarch -batch -ex 'file <.elf file>' -ex 'disassemble /mr <function>'
  ```