- Step 1
- Step 2: Relocation
  .elf -> .img
    - use objcopy for bytes
    - re-organize to become .img file that can be loaded into machine

Kernel Mode
- execution w/ full privileges of the hardware
- read/write to memory, access I/O device, send/read network packet

User Mode
- limited privileges

- 2 hats

- compilation does not care about privileges

- hardware that periodically interrupts processor
- hardware timer => controls interrupts
  periodically interrupts the processor
- kernel must copy string from user program to screen memory

- interrupts: asynchronous "irq"
  ↳ external event happened

- exception: synchronous
  ↳ one-time events
  ↳ programming errors

Hardware Support for OS

- RPI3: gpu focused
- 64-bit v. 32-bit
  - represented in bytes
  - address

Exception Levels

EL0 Applications ← EL1 Rich OS ← EL2 Hypervisor ← EL3 Firmware

permission required

- registers = 64-bit
- currentEL ← only accessible EL1+

EL0 App → Usr
EL1 Rich OS
EL2 Hypervisor
EL3 Firmware
kernel

configure registers

- when exception happens (eret)
  - current processor state to SPSR
  - save return address to ELR
- irq => exception handler; handles things like timers

b: short jump; relative address

br: long jump; address stored in register

PState

- condition flags
- interrupt masks: D, A, I, F
- execution state

mrs: moves system register to general register

msr: write register to system register

- Pstate —> SPSR
- SPSR_ELx ⇐ saved processor state
- ELR_ELx: saved PC upon taking exception
  - nature of instruction determines if current or future address/instruction that is held
- boots from highest exception level