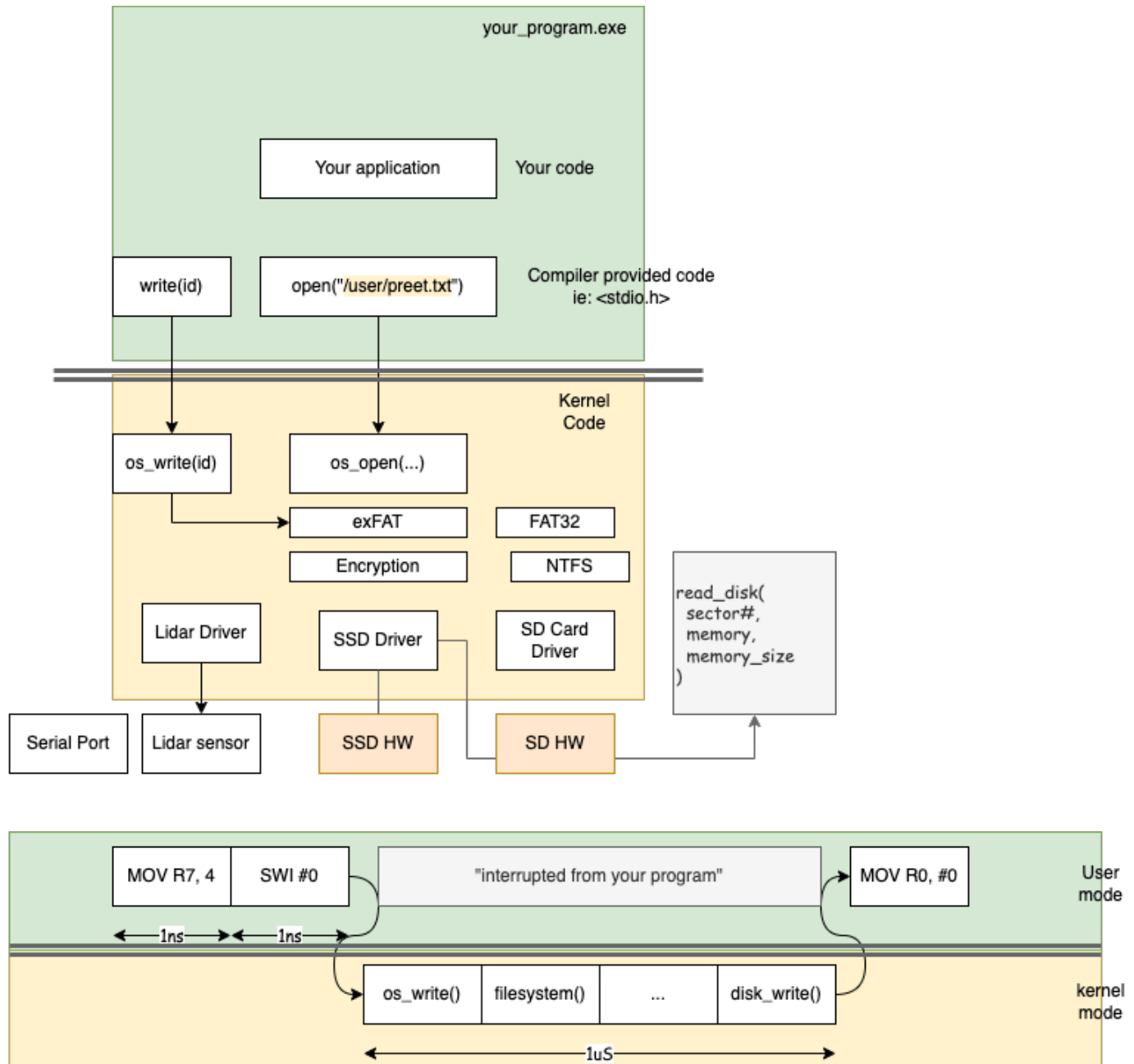


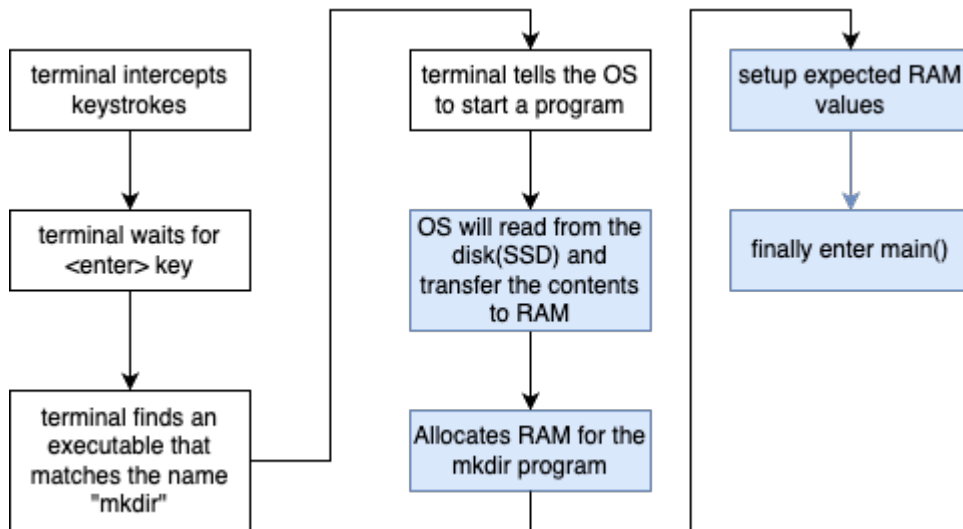
# Operating System

- [System Call](#)
- [How OS launches a program](#)
- [Fundamentals of an OS](#)

# System Call



# How OS launches a program



# Fundamentals of an OS

## What is an OS?

A good example can be reference at section 6.5 of [Structured Computer Organization](#).

Fundamentally, an OS provides services:

- File services (open, read, write, close etc.)
- Networking services (open, read, write, close internet socket)
  - Berkeley socket interface
- Multi-tasking services
  - Creating multiple threads
  - Dictating priorities for threads

## Interrupt

An interrupt is an asynchronous function call that can interrupt normal flow of a program.

```

// HW and OS work together to invoke this function
// whenever a keyboard key is pressed
// Regardless of where you are at your_program()
// this function can be invoked asynchronously
void interrupt_keyboard(void) {
}

// Thread that never exits
// This calls all the sub-functions synchronously
void your_program(void) {
    while (forever) {
        check_for_brake_pedal();
        actuate_brakes();
    }
}

```

## Kernel vs. User Space

```

// OS use "SWI" or "Software Interrupt"
// "Software Interrupt" really means a "Deliberate Interrupt request to the HW"
void deliberate_interrupt(void) {
}

//
void your_program(void) {
    while (forever) {
        int file = open("file.txt");
    }
}

// Psuedo-code for open
// This is what the open function looks like inside the OS
void open(filename) {
    R4 = open_request_number;
    R5 = filename
    SWI
}

```

# Virtual Memory

Virtual memory is not purely virtual memory, it is virtual memory addresses that maps to real and physical hardware memory.

## POSIX Interface