src/cpu/**mod.rs**

```
let byte = self.memory.read_byte(self.registers.pc);
self.registers.pc = self.registers.pc.wrapping_add(1);
let ticks = match byte {
    0x00 => 1,
    0x01 => { ld::bcnn(self); 3 }
    0x02 => { ld::bcm_a(self); 2 }
    0x03 => { data::incbc(self); 2 }

    // … (rest of the instructions)

    0xFF => { stack::rst(self, 0x38); 4 }
    _ => { panic!("{:#06x} not implemented", op);
}
self.memory.cycle(ticks * 4);
```

Interrupts

Interrupt register is just an 8 bit value consisting of flags (single bits) to indicate what kind of interrupts are enabled.

It interrupt the current program flow in response to certain events.