

Foreword

Acknowledgements

History

Overview

1. Specifications

2. Memory Map

I/O Ports

3. Rendering

4. Sound Controller

5. Joypad Input

6. Serial Data Transfer

7. Timer and Divider Registers

8. Interrupts

8.1. Interrupt Sources

8.2. HALT

9. CGB Registers

10. Infrared Communication

11. SGB Functions

CPU Specifications

12. CPU Registers and Flags

halt

halt is an instruction that pauses the CPU (during which less power is consumed) when executed. The CPU wakes up as soon as an interrupt is pending, that is, when the bitwise AND of `IE` and `IF` is non-zero.

Most commonly, `IME` is set. In this case, the CPU simply wakes up, and before executing the instruction after the halt, the `interrupt handler` is called normally.

If `IME` is *not* set, there are two distinct cases, depending on whether an interrupt is pending as the halt instruction is first executed.

- If no interrupt is pending, halt executes as normal, and the CPU resumes regular execution as soon as an interrupt becomes pending. However, since `IME=0`, the interrupt is not handled.
- If an interrupt is pending, halt immediately exits, as expected, however the “halt bug” explained below, is triggered.

halt bug

Under some circumstances, pc fails to be normally incremented.

The most typical trigger, halt with `IME=0` and `[IE] & [IF] != 0`, causes the byte after the halt to be read twice.

The behavior is different when `ei` (whose effect is typically delayed by one instruction) is followed immediately by a halt, and an interrupt is pending as the halt is executed. The interrupt is serviced and the handler called, but the interrupt returns to the halt, which is executed again, and thus waits for another interrupt. (Source)