Computing Power is Everywhere

Rechenleistung, „Dinge die etwas berechnen können“, muss nicht in klassischen Chips passieren. Historisch: Siehe etwa Babbage's nie realisierte Analytical Engine oder Zuse's Z1. Solche Maschinen sind nicht Turing complete (z. B. keine Speichermöglichkeit) aber *functional complete*. Zeitgemäße Spielereien: 4-Bit Rechner mit Dominos oder Wasser (greedy cup) sind realisierbar.

Aber auch „Maschinen“ die *turing complete* sind müssen nicht wie ein klassischer Computer aussehen. So können etwa durch die Beziehungen und wechselseitigen Effekte zwischen Magic: The Gathering Karten (und einem Tape aus umfunktionierten Karten) alle notwendigen Features einer Turingmaschine realisert werden. (Weitere offensichtlichere Beispiele: Alle Spiele mit funktionierenden logic gates: Factorio, Terraria, Shenzhen, . . . , Minecraft?)

Minecraft

Turing Machine Realisierung einer DFA ist durch wechselseitige Steuerung von minecarts und redstone relativ trivial möglich. *Transition matrix* realisiert durch Schienennetz, ein minecart (vgl. %rip) kann alle states erreichen. Das Stellwerk bekommt input und stellt die Schienen so ein, dass das minecart die entsprechenden state transitions durchmacht.

Der tape head ist ebenfalls ein minecart, das durch redstone Ansteuerung bzw. Schienenstellung immer eine Stelle nach links oder rechts bewegt werden kann. Auf den Ruheplätzen sind Druckplatten die die aktuelle Position abrufbar machen.

Computer? Alle Logikgatter können mit redstone umgesetzt werden, also ist eine ALU etc. möglich. Arbeitsspeicher ist durch ein paar Tricks auch umsetzbar und überraschend gut skalierbar. Also können alle notwendigen Teile einer recht simplistischen CPU realisiert werden.

Aber: Control units sind kompliziert, ein ordentliches instruction set zu implementieren ist nontrivial, glücklicherweise reicht hier ein recht vereinfachtes. Die control unit kann in etwa wie eine DFA aufgebaut werden.