Addressing Memory (Pt1).

In our previous video on how memory worked we got to the point from storing one bit of information that we could set or reset it to the flip flop where we could store a bit of info and then we arranged for them together in parallel to store four bits of information so what we did we had four inputs d naught 2 d 3 which gave us our input and four outputs q naught to q 3. CPU refers to every single memory location by a unique binary number. Now I want to be able to access either this one or this one. So, I need this thing to be 1 whenever A in is 0. The NAND gate has two inputs, a and b, and an output, which again will use q. whenever a is 1 the output matches b. When the input to the knock cage is 0, the output is 1. When the input is not gate is 1, so we need this thing to be 1 whenever a in is 0.

Archimedes would run reliably above 100 degrees Celsius this is just the half adder we want a binary full. We can choose whichever flip flop we want by the address and then we can strobe the w thing to write it to that one or that one.

Total Recall (Memory Addressing Pt2).

We can choose whichever flip flop we want by the address and then we can strobe the w thing to write it to that one or that one. But what about reading. How do we combine the two outputs from our flip flops into a single output which we'll call d out here? I'm going to move this paper, so will let me just copy that quickly. We have our two inputs A and B and our output q. We need to combine these two together to produce our output. The way we do that is by using an OR gate looking at the inputs like that. You could build that out using many address lines and many all gates around gates to produce your whole memory. We have a grid effectively of cells where we can store our data , and then again we have the multiplexer is using all gates to combine them together to produce output and then we can stack these in 3d to produce the sort of 8bit outputs like I've got here.