qRAM and Uncomputation

Consider encoding data in qubits. Assume the following array.

```
a = ▸Int64[0, 0, 1, 0, 0]
```

```
a = [0, 0, 1, 0, 0]
```

It'll take 5 qubits to encode this.

```
using Yao, YaoPlots
```

```
▸BitBasis.BitStr{5,Int64}[00100 ₍₂₎]
```

```
ArrayReg(bit"00100") |> r->measure(r)
```

or

```
▸BitBasis.BitStr{5,Int64}[00100 ₍₂₎]
```

```
zero_state(5) |> put(5, 3=>X) |> r->measure(r)
```

Either way, lets assume we've to encode 4 such arrays, in qubits.

```
b = ▸Int64[0, 1, 1, 0, 1]
```

```
b = [0, 1, 1, 0, 1]
```

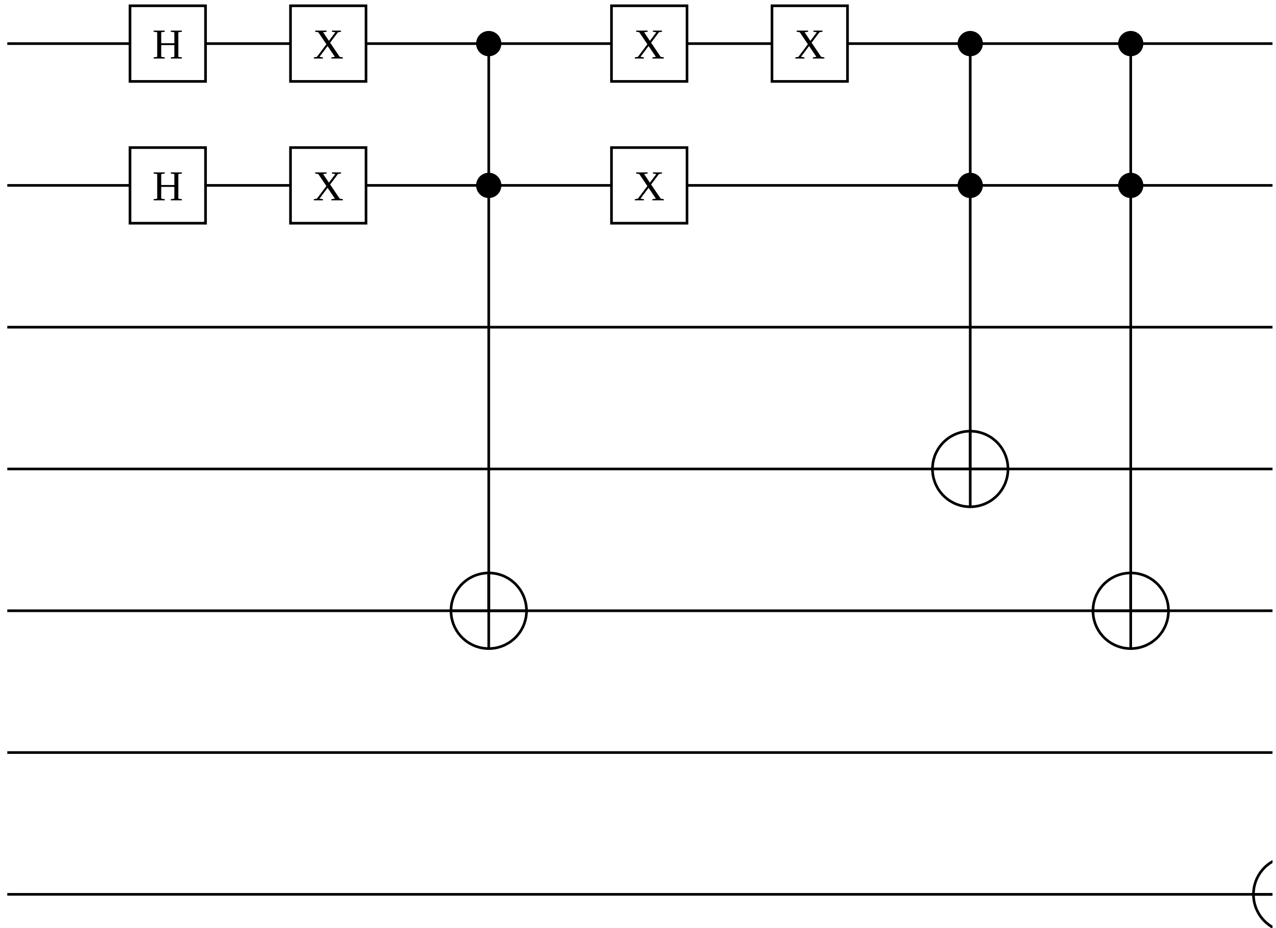
```
c = ▸Int64[1, 1, 0, 0, 0]
```

```
c = [1, 1, 0, 0, 0]
```

```
d = ▸Int64[1, 0, 1, 1, 1]
```

```
d = [1, 0, 1, 1, 1]
```

To encode these 4 arrays, it'd take 20 qubits, judging by the above approach. But using QRAMS, we can use 7 qubits, to encode all the 4 arrays.




```
let
    f(x) = chain(7, [control(1:2, (k+2)=>X) for k in findall(isone, x)])
    global QRAM = chain(7, repeat(H, 1:2), repeat(X, 1:2), f(a), repeat(X, 1:2),
put(1=>X), f(b), put(1=>X), put(2=>X), f(c), put(2=>X), f(d))
    plot(QRAM)
end
```

The first two qubits, are called the address qubits.

- When the address qubits give 00 or 0 in decimal, for the next 5 qubits, we get 00100.
- When the address qubits give 01 or 1 in decimal, for the next 5 qubits, we get 01101.
- When the address qubits give 10 or 2 in decimal, for the next 5 qubits, we get 11000.
- When the address qubits give 11 or 3 in decimal, for the next 5 qubits, we get 10111.

The input to the QRAM is 0000000.

```
output =
▶BitBasis.BitStr{7,Int64}[1110111 ₍₂₎,  1011010 ₍₂₎,  1110111 ₍₂₎,  1110111 ₍₂₎,  0001101 ₍
```

```
output = zero_state(7) |> QRAM |> r->measure(r, nshots = 1024)
```

The below code records the frequency of measurements.

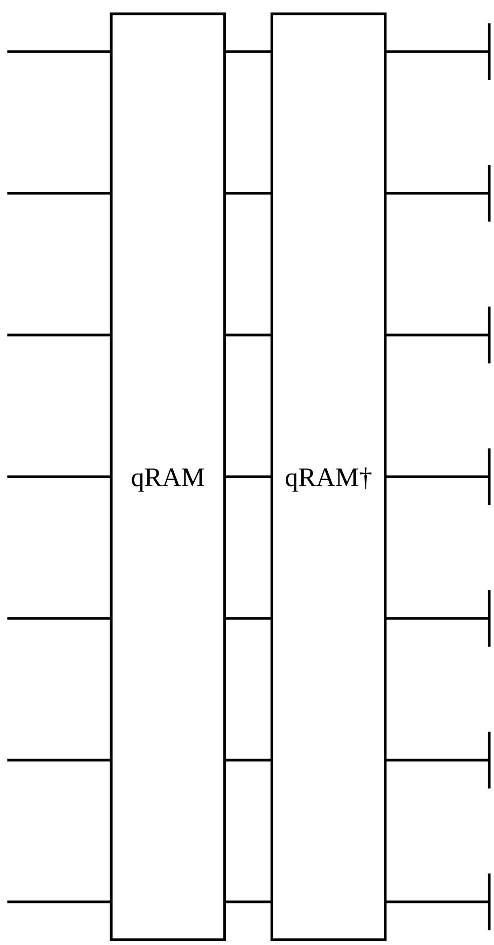
```
String["1011000", "0000100", "0101101", "1110111"]
```

```
begin
    using StatsBase: fit, Histogram
    hist = fit(Histogram, Int.(output), 0:2^7)
    o1 = hist.weights[findall(!iszero, hist.weights)]
    o2 = reverse.(string.(0:(2^7-1), base=2, pad=7)[findall(!iszero, hist.weights)])
end
```

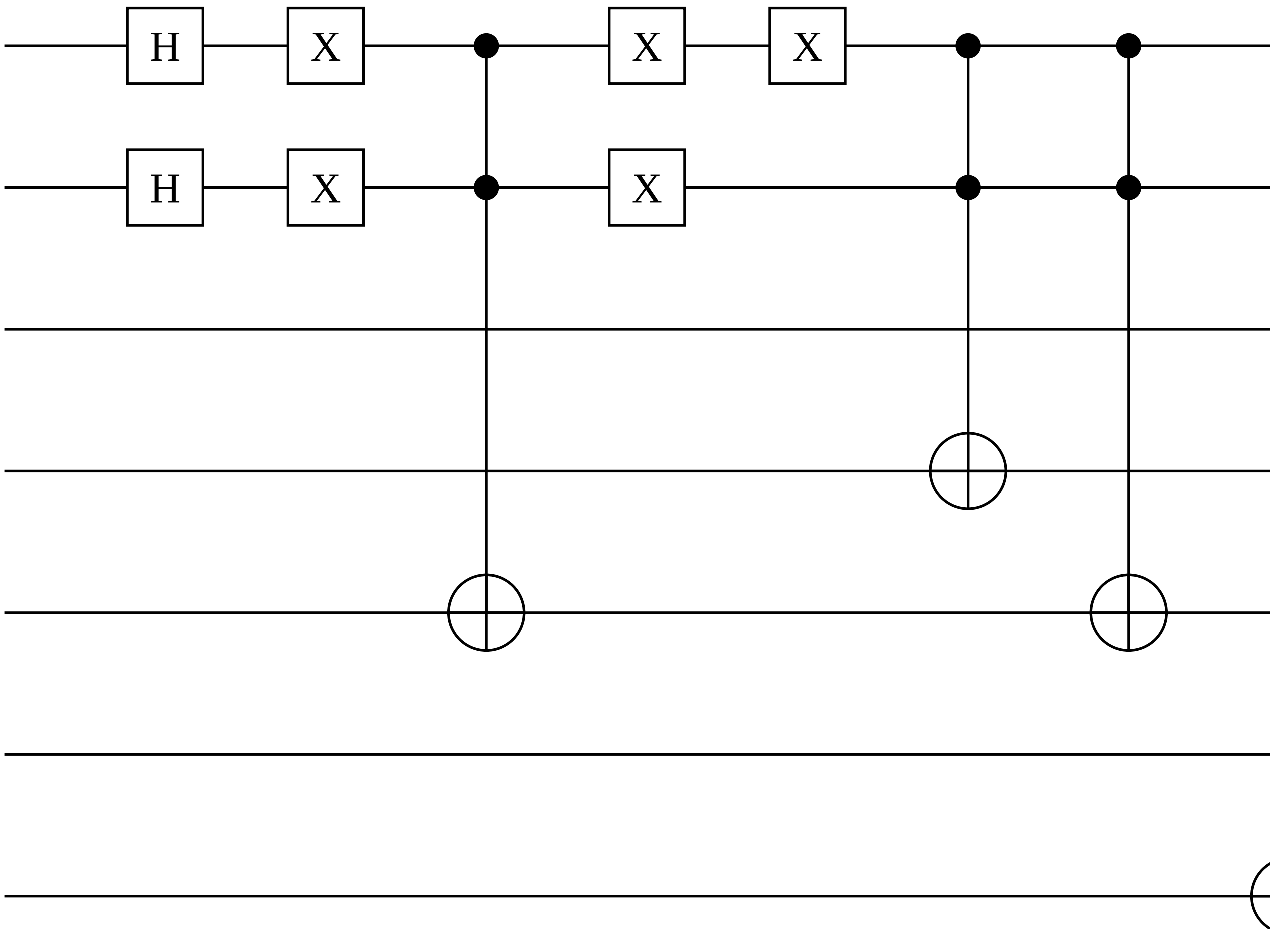
- When address qubits were 10, we got 11000, with the frequency 251.
- When address qubits were 00, we got 00100, with the frequency 240.
- When address qubits were 01, we got 01101, with the frequency 236.
- When address qubits were 11, we got 10111, with the frequency 297.

We use Uncomputation to reverse the everything we do in a circuit to reverse it to its former state. It's often used with qRAMs.

An example would be



```
plot(chain(7, put(1:7 => label(QRAM,"qRAM")), put(1:7 =>
label(Daggered(QRAM),"qRAM†"))))
```




```
begin
    uncomputation = chain(7, put(1:7 => QRAM), put(1:7 => QRAM'))
    plot(uncomputation)
end
```

```
▶ BitBasis.BitStr{7,Int64}[0000000 ₍₂₎, 0000000 ₍₂₎, 0000000 ₍₂₎, 0000000 ₍₂₎, 0000000 ₍
```

```
zero_state(7) |> uncomputation |> r->measure(r, nshots=1024)
```

As you can see, we first apply the QRAM circuit, and then its dagger, which undoes the effect.