```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/03/a/Bit.hdl

/**
 * 1-bit register:
 * If load[t] == 1 then out[t+1] = in[t]
 *                 else out does not change (out[t+1] = out[t])
 */

CHIP Bit {
    IN in, load;
    OUT out;

    PARTS:
    Mux( a = loop, b = in, sel = load, out = w1);
    DFF( in = w1, out = out, out = loop);
}
```

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/03/a/Register.hdl

/**
 * 16-bit register:
 * If load[t] == 1 then out[t+1] = in[t]
 * else out does not change
 */

CHIP Register {
    IN in[16], load;
    OUT out[16];

    PARTS:
    Bit( in = in[0], load = load, out = out[0]);
    Bit( in = in[1], load = load, out = out[1]);
    Bit( in = in[2], load = load, out = out[2]);
    Bit( in = in[3], load = load, out = out[3]);
    Bit( in = in[4], load = load, out = out[4]);
    Bit( in = in[5], load = load, out = out[5]);
    Bit( in = in[6], load = load, out = out[6]);
    Bit( in = in[7], load = load, out = out[7]);
    Bit( in = in[8], load = load, out = out[8]);
    Bit( in = in[9], load = load, out = out[9]);
    Bit( in = in[10], load = load, out = out[10]);
    Bit( in = in[11], load = load, out = out[11]);
    Bit( in = in[12], load = load, out = out[12]);
    Bit( in = in[13], load = load, out = out[13]);
    Bit( in = in[14], load = load, out = out[14]);
    Bit( in = in[15], load = load, out = out[15]);
}
```

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/03/a/RAM8.hdl

/**
 * Memory of 8 registers, each 16 bit-wide. Out holds the value
 * stored at the memory location specified by address. If load==1, then
 * the in value is loaded into the memory location specified by address
 * (the loaded value will be emitted to out from the next time step onward).
 */

CHIP RAM8 {
    IN in[16], load, address[3];
    OUT out[16];

    PARTS:
    DMux8Way( in = load, sel = address, a = a1, b = b1, c = c1, d = d1, e = e1,
            f = f1, g = g1, h = h1);
    Register( in = in, load = a1, out = w1);
    Register( in = in, load = b1, out = w2);
    Register( in = in, load = c1, out = w3);
    Register( in = in, load = d1, out = w4);
    Register( in = in, load = e1, out = w5);
    Register( in = in, load = f1, out = w6);
    Register( in = in, load = g1, out = w7);
    Register( in = in, load = h1, out = w8);
    Mux8Way16( a = w1, b = w2, c = w3, d = w4, e = w5, f = w6, g = w7, h = w8,
            sel = address, out = out);
}
```

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/03/a/RAM64.hdl

/**
 * Memory of 64 registers, each 16 bit-wide. Out holds the value
 * stored at the memory location specified by address. If load==1, then
 * the in value is loaded into the memory location specified by address
 * (the loaded value will be emitted to out from the next time step onward).
 */

CHIP RAM64 {
    IN in[16], load, address[6];
    OUT out[16];

    PARTS:
    DMux8Way( in = load, sel = address[3..5], a = a1, b = b1, c = c1, d = d1, e = e1,
            f = f1, g = g1, h = h1);
    RAM8( in = in, load = a1, address = address[0..2], out = w1);
    RAM8( in = in, load = b1, address = address[0..2], out = w2);
    RAM8( in = in, load = c1, address = address[0..2], out = w3);
    RAM8( in = in, load = d1, address = address[0..2], out = w4);
    RAM8( in = in, load = e1, address = address[0..2], out = w5);
    RAM8( in = in, load = f1, address = address[0..2], out = w6);
    RAM8( in = in, load = g1, address = address[0..2], out = w7);
    RAM8( in = in, load = h1, address = address[0..2], out = w8);
    Mux8Way16( a = w1, b = w2, c = w3, d = w4, e = w5, f = w6, g = w7, h = w8,
            sel = address[3..5], out = out);
}
```

```
// This file is part of the materials accompanying the book
// "The Elements of Computing Systems" by Nisan and Schocken,
// MIT Press. Book site: www.idc.ac.il/tecs
// File name: projects/03/b/RAM512.hdl

/**
 * Memory of 512 registers, each 16 bit-wide. Out holds the value
 * stored at the memory location specified by address. If load==1, then
 * the in value is loaded into the memory location specified by address
 * (the loaded value will be emitted to out from the next time step onward).
 */

CHIP RAM512 {
    IN in[16], load, address[9];
    OUT out[16];

    PARTS:
    DMux8Way( in = load, sel = address[6..8], a = a1, b = b1, c = c1, d = d1, e = e1,
            f = f1, g = g1, h = h1);
    RAM64( in = in, load = a1, address = address[0..5], out = w1);
    RAM64( in = in, load = b1, address = address[0..5], out = w2);
    RAM64( in = in, load = c1, address = address[0..5], out = w3);
    RAM64( in = in, load = d1, address = address[0..5], out = w4);
    RAM64( in = in, load = e1, address = address[0..5], out = w5);
    RAM64( in = in, load = f1, address = address[0..5], out = w6);
    RAM64( in = in, load = g1, address = address[0..5], out = w7);
    RAM64( in = in, load = h1, address = address[0..5], out = w8);
    Mux8Way16( a = w1, b = w2, c = w3, d = w4, e = w5, f = w6, g = w7, h = w8,
            sel = address[6..8], out = out);
}
```

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/03/b/RAM4K.hdl

/**
 * Memory of 4K registers, each 16 bit-wide. Out holds the value
 * stored at the memory location specified by address. If load==1, then
 * the in value is loaded into the memory location specified by address
 * (the loaded value will be emitted to out from the next time step onward).
 */

CHIP RAM4K {
    IN in[16], load, address[12];
    OUT out[16];

    PARTS:
    DMux8Way( in = load, sel = address[9..11], a = a1, b = b1, c = c1, d = d1, e = e1,
            f = f1, g = g1, h = h1);
    RAM512( in = in, load = a1, address = address[0..8], out = w1);
    RAM512( in = in, load = b1, address = address[0..8], out = w2);
    RAM512( in = in, load = c1, address = address[0..8], out = w3);
    RAM512( in = in, load = d1, address = address[0..8], out = w4);
    RAM512( in = in, load = e1, address = address[0..8], out = w5);
    RAM512( in = in, load = f1, address = address[0..8], out = w6);
    RAM512( in = in, load = g1, address = address[0..8], out = w7);
    RAM512( in = in, load = h1, address = address[0..8], out = w8);
    Mux8Way16( a = w1, b = w2, c = w3, d = w4, e = w5, f = w6, g = w7, h = w8,
            sel = address[9..11], out = out);
}
```

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/03/b/RAM16K.hdl

/**
 * Memory of 16K registers, each 16 bit-wide. Out holds the value
 * stored at the memory location specified by address. If load==1, then
 * the in value is loaded into the memory location specified by address
 * (the loaded value will be emitted to out from the next time step onward).
 */

CHIP RAM16K {
    IN in[16], load, address[14];
    OUT out[16];

    PARTS:
    DMux4Way( in = load, sel = address[12..13], a = a1, b = b1, c = c1, d = d1);
    RAM4K( in = in, load = a1, address = address[0..11], out = w1);
    RAM4K( in = in, load = b1, address = address[0..11], out = w2);
    RAM4K( in = in, load = c1, address = address[0..11], out = w3);
    RAM4K( in = in, load = d1, address = address[0..11], out = w4);
    Mux4Way16( a = w1, b = w2, c = w3, d = w4, sel = address[12..13], out = out);
}
```

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/03/a/PC.hdl

/**
 * A 16-bit counter with load and reset control bits.
 * if      (reset[t] == 1) out[t+1] = 0
 * else if (load[t] == 1)  out[t+1] = in[t]
 * else if (inc[t] == 1)   out[t+1] = out[t] + 1  (integer addition)
 * else                    out[t+1] = out[t]
 */

CHIP PC {
    IN in[16],load,inc,reset;
    OUT out[16];

    PARTS:
    Inc16( in = loop, out = w1);
    Mux16( a = loop, b = w1, sel = inc, out = w2);
    Mux16( a = w1, b = in, sel = load, out = w3);
    Mux16( a = w3, b = false, sel = reset, out = w4);

    Or( a = reset, b = load, out = x1);
    Or( a = x1, b = inc, out = x2);
    Register( in = w4, load = x2, out = out, out = loop);
}
```