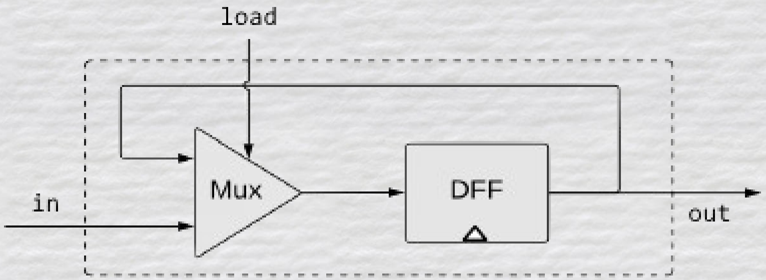
Computer Organization

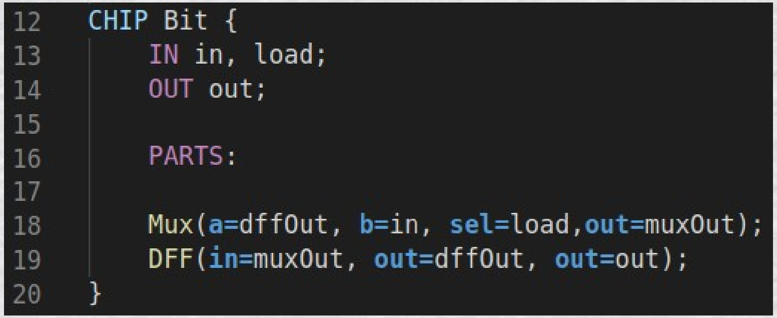
Sequential Gates

1-Bit Register (Bit)

- The simplest form of sequential logic.
  - Simply uses a Mux and a DFF (data flip-flop)



- DFF loop back into the Mux using a clock
  - DFF is both the output of the gate and part of the input



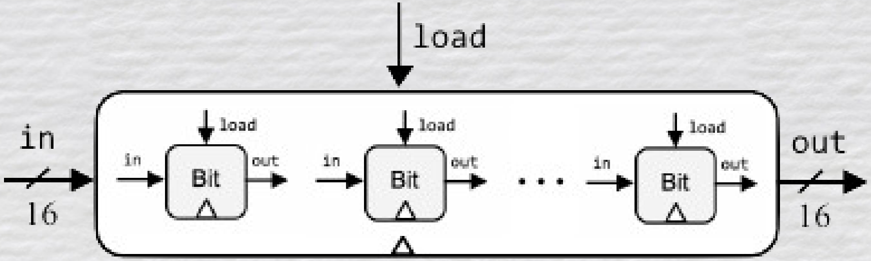
```
CHIP Bit {
    IN in, load;
    OUT out;

    PARTS:

    Mux(a=dffOut, b=in, sel=load,out=muxOut);
    DFF(in=muxOut, out=dffOut, out=out);
}
```

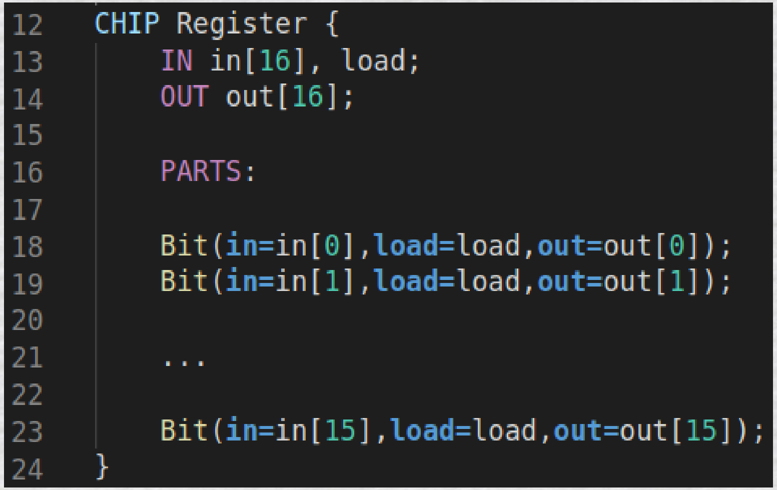
16-Bit Register (Register)

- Very similar to the previous, but handles 16-bit sequences
  - You've seen many of these so far…

    Now it's just sequential

- Multiple of these will be used to create RAM

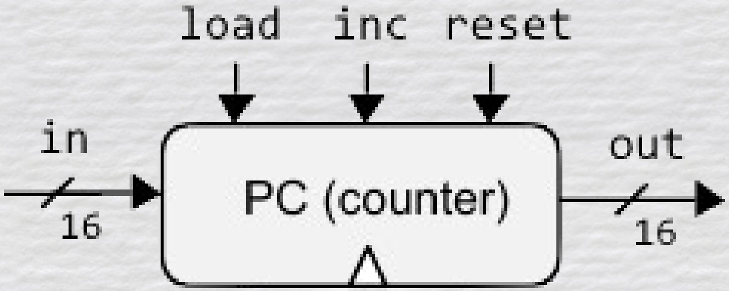


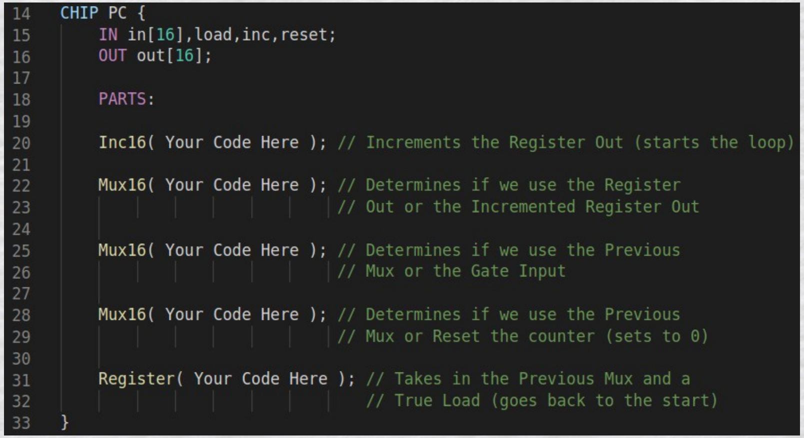


```
CHIP Register {
    IN in[16], load;
    OUT out[16];

    PARTS:

    Bit(in=in[0],load=load,out=out[0]);
    Bit(in=in[1],load=load,out=out[1]);

    ...

    Bit(in=in[15],load=load,out=out[15]);
}
```

16-Bit Counter (PC)

- Used to navigate our future assembly code
  1. Chooses to either select our current instruction or increment to the next
  2. Chooses between the former and the input to our gate (used to jump)
  3. Chooses between the former and **false** (used to reset to the beginning)

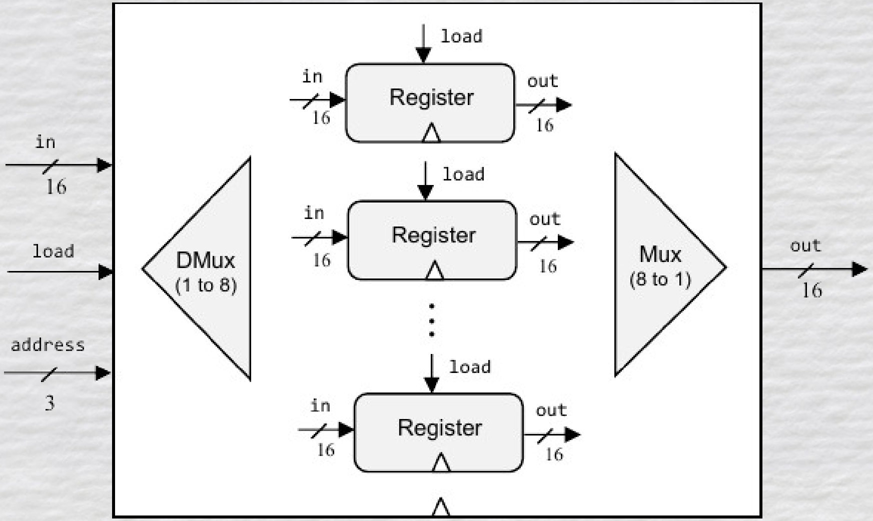


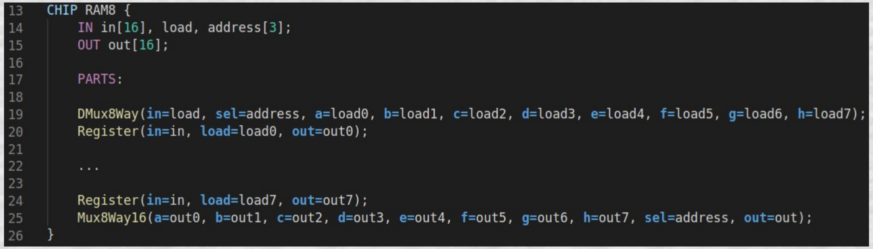


```
CHIP PC {
    IN in[16],load,inc,reset;
    OUT out[16];

    PARTS:

    Inc16( Your Code Here ); // Increments the Register Out (starts the loop)

    Mux16( Your Code Here ); // Determines if we use the Register
                             // Out or the Incremented Register Out

    Mux16( Your Code Here ); // Determines if we use the Previous
                             // Mux or the Gate Input

    Mux16( Your Code Here ); // Determines if we use the Previous
                             // Mux or Reset the counter (sets to 0)

    Register( Your Code Here ); // Takes in the Previous Mux and a
                                // True Load (goes back to the start)
}
```

RAM

- All of the RAM chips work on a very similar principal
  - It is a decision of what Register to store data in (DMux → Registers → Mux)
- We start with RAM8 which is just a collection of Registers that all share a load and 16-bit sequence

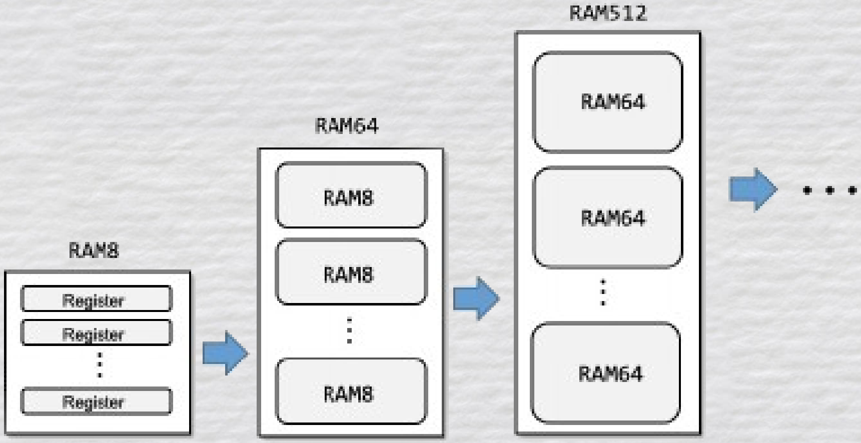


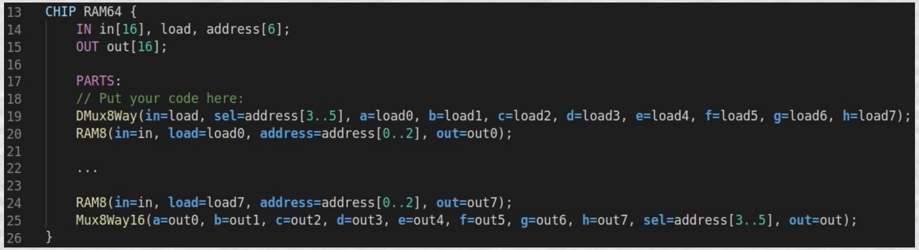


```
CHIP RAM8 {
    IN in[16], load, address[3];
    OUT out[16];

    PARTS:

    DMux8Way(in=load, sel=address, a=load0, b=load1, c=load2, d=load3, e=load4, f=load5, g=load6, h=load7);
    Register(in=in, load=load0, out=out0);

    ...

    Register(in=in, load=load7, out=out7);
    Mux8Way16(a=out0, b=out1, c=out2, d=out3, e=out4, f=out5, g=out6, h=out7, sel=address, out=out);
}
```

RAM – cont.

- From there, RAM just builds as a collection of previous RAM chips.
  - RAM64 is just 8 RAM8 chips
  - RAM512 is just 8 RAM64 chips
    - These use an 8-Way DMux and Mux
  - RAM16K is just 4 RAM4K chips
    - These use a 4-Way DMux and Mux





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

    PARTS:
    // Put your code here:
    DMux8Way(in=load, sel=address[3..5], a=load0, b=load1, c=load2, d=load3, e=load4, f=load5, g=load6, h=load7);
    RAM8(in=in, load=load0, address=address[0..2], out=out0);

    ...

    RAM8(in=in, load=load7, address=address[0..2], out=out7);
    Mux8Way16(a=out0, b=out1, c=out2, d=out3, e=out4, f=out5, g=out6, h=out7, sel=address[3..5], out=out);
}
```