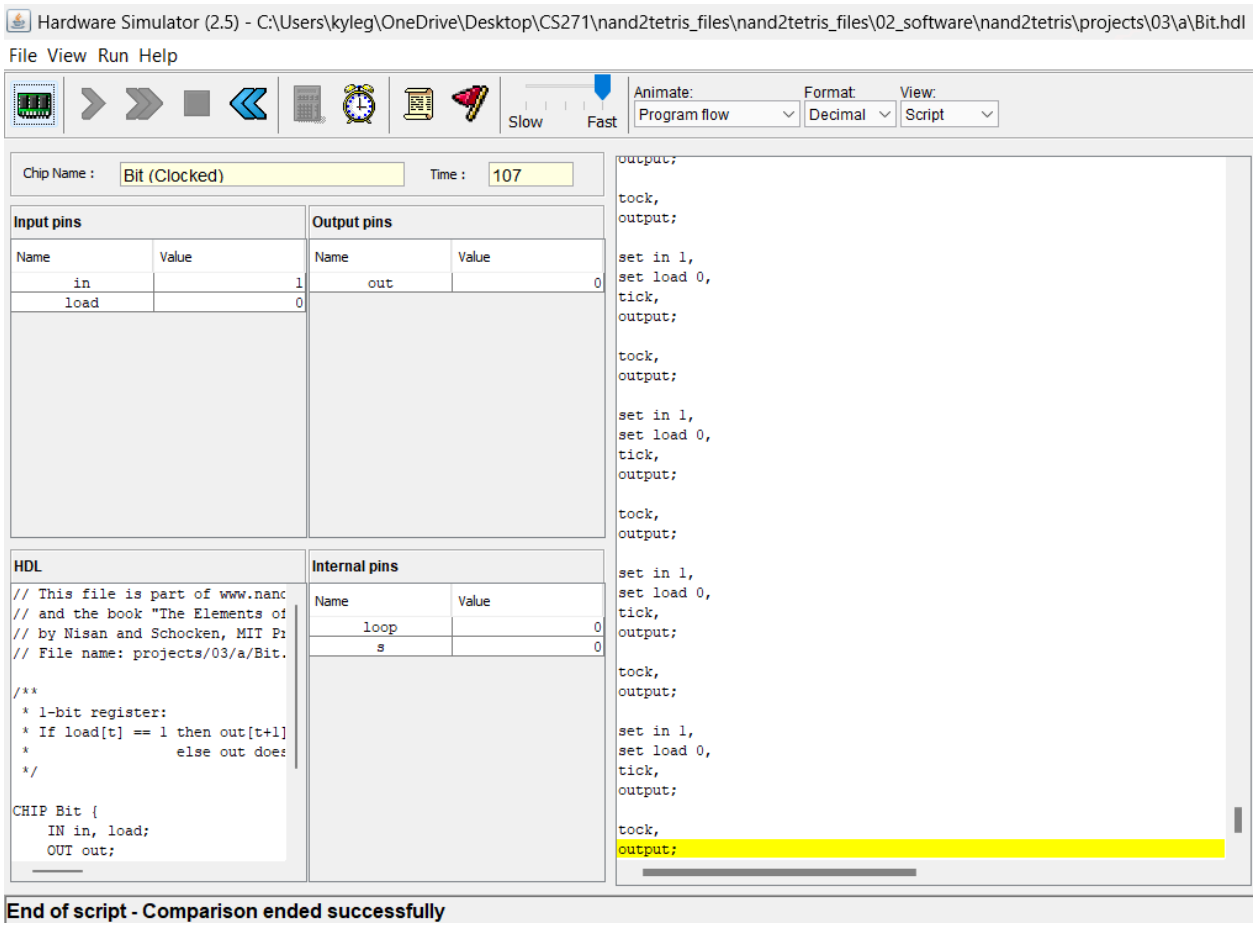
BIT CHIP:

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

    PARTS:
    // Use the Muxer to repeat values when load in ON and use DFF to relay
previous
    Mux(a=loop, b=in, sel=load, out=s);
    DFF(in=s, out=out, out=loop);
}
```

BIT TEST:

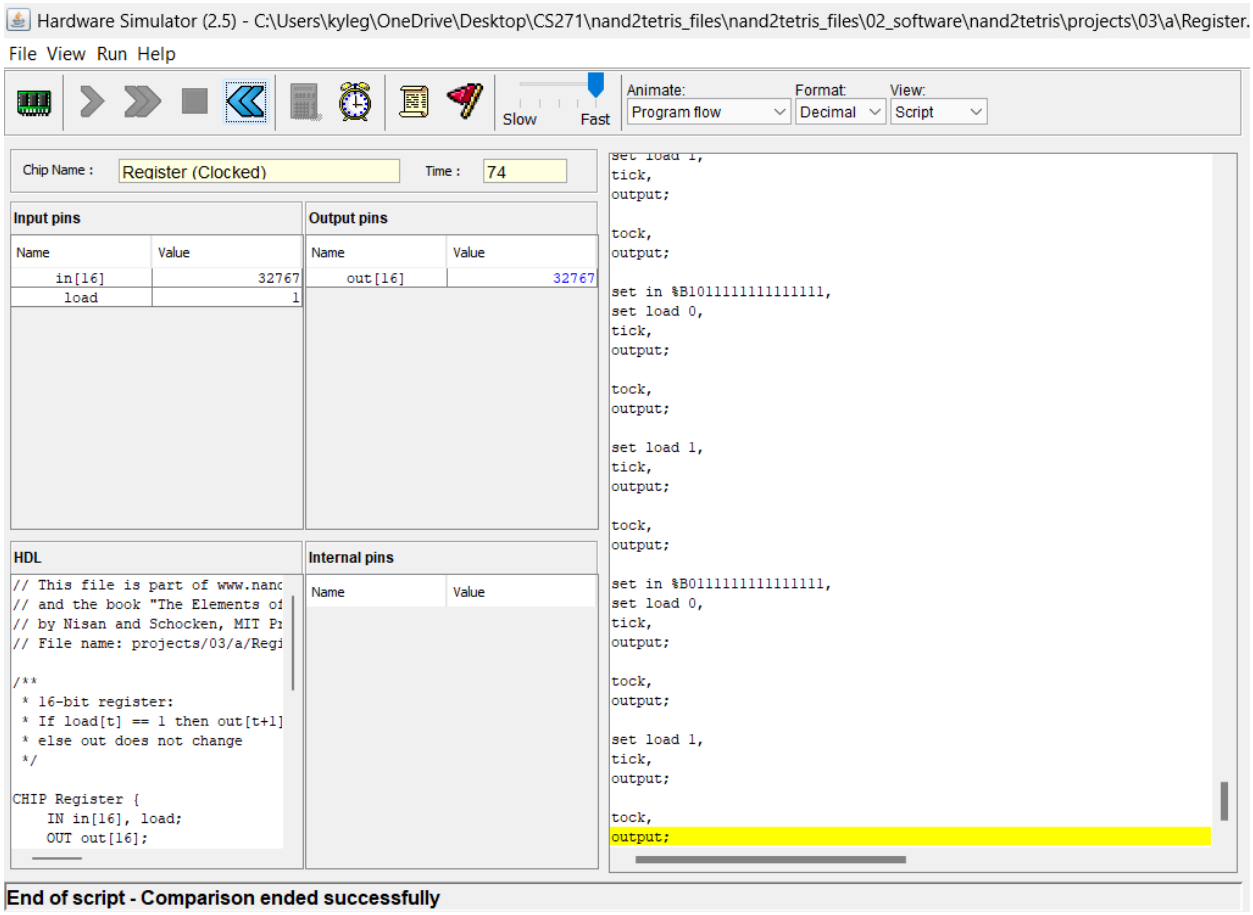


REGISTER CHIP:

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

    PARTS:
    // 16 bit storer
    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]);
}
```

REGISTER TEST:



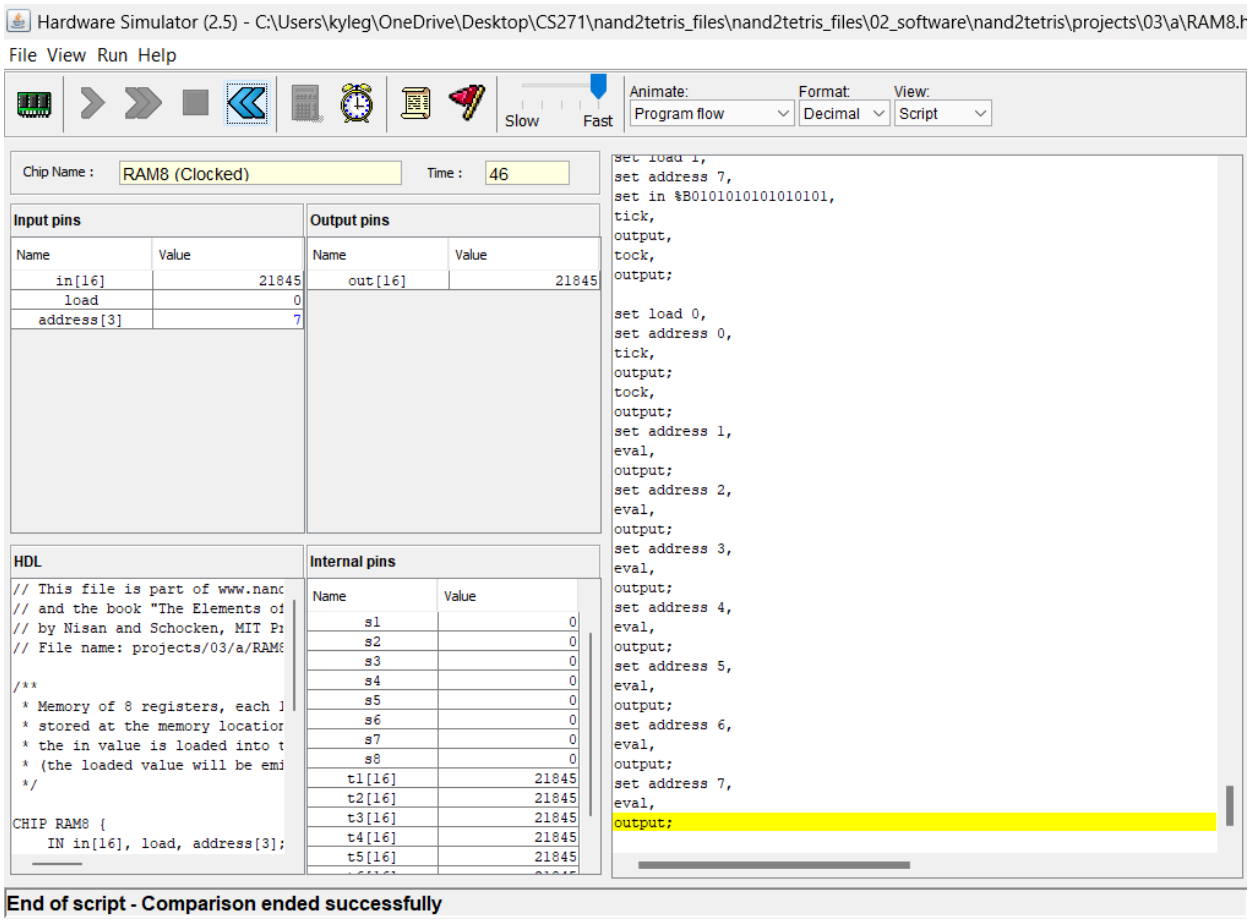
RAM8 CHIP:

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

    PARTS:
    // Putting 8 registers together that can be selected
    DMux8Way(in=load, sel=address, a=s1, b=s2, c=s3, d=s4, e=s5, f=s6,
g=s7, h=s8);
    Register(in=in, load=s1, out=t1);
    Register(in=in, load=s2, out=t2);
    Register(in=in, load=s3, out=t3);
    Register(in=in, load=s4, out=t4);
    Register(in=in, load=s5, out=t5);
    Register(in=in, load=s6, out=t6);
    Register(in=in, load=s7, out=t7);
    Register(in=in, load=s8, out=t8);
```

```
    Mux8Way16(a=t1, b=t2, c=t3, d=t4, e=t5, f=t6, g=t7, h=t8, sel=address,
out=out);
}
```

RAM8 TEST:



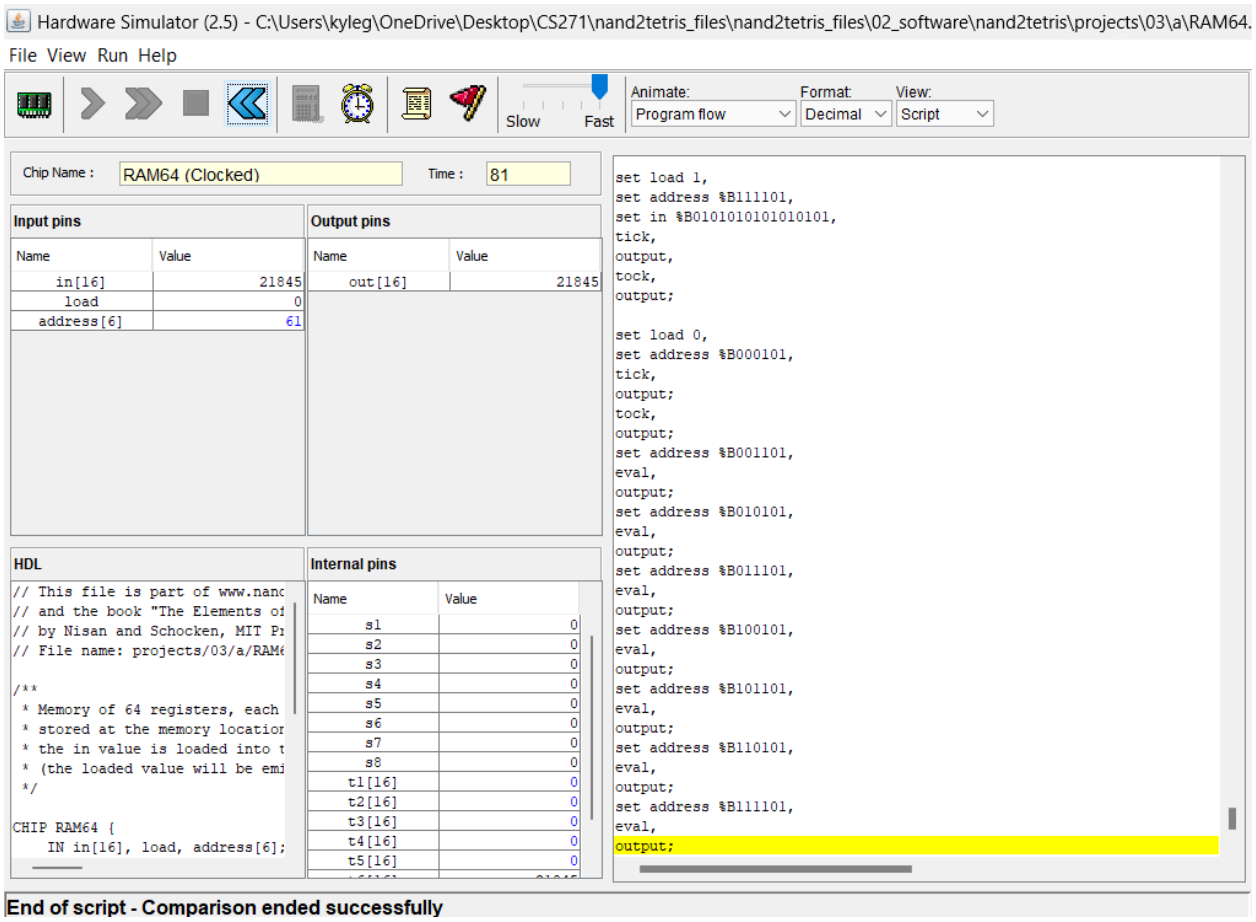
RAM64 CHIP:

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

    PARTS:
    // Putting 8 8-bit RAMs together
    DMux8Way(in=load, sel=address[0..2], a=s1, b=s2, c=s3, d=s4, e=s5,
f=s6, g=s7, h=s8);
    RAM8(in=in, load=s1, address=address[3..5], out=t1);
    RAM8(in=in, load=s2, address=address[3..5], out=t2);
    RAM8(in=in, load=s3, address=address[3..5], out=t3);
```

```
    RAM8(in=in, load=s4, address=address[3..5], out=t4);
    RAM8(in=in, load=s5, address=address[3..5], out=t5);
    RAM8(in=in, load=s6, address=address[3..5], out=t6);
    RAM8(in=in, load=s7, address=address[3..5], out=t7);
    RAM8(in=in, load=s8, address=address[3..5], out=t8);
    Mux8Way16(a=t1, b=t2, c=t3, d=t4, e=t5, f=t6, g=t7, h=t8,
sel=address[0..2], out=out);
}
```

RAM64 TEST:



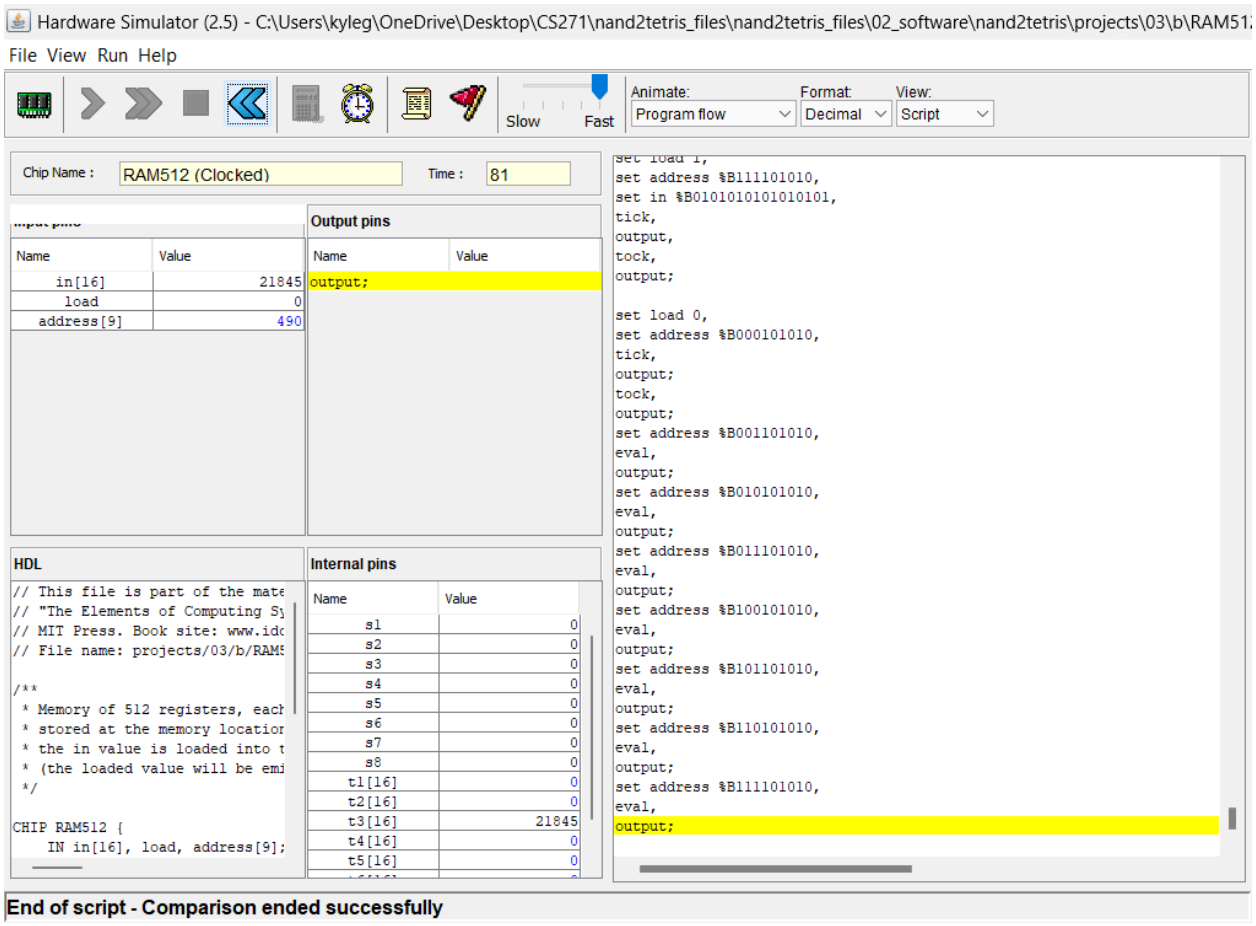
RAM512 CHIP:

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

    PARTS:
    // Putting 8 64-bit RAMs together
```

```
    DMux8Way(in=load, sel=address[0..2], a=s1, b=s2, c=s3, d=s4, e=s5,
f=s6, g=s7, h=s8);
    RAM64(in=in, load=s1, address=address[3..8], out=t1);
    RAM64(in=in, load=s2, address=address[3..8], out=t2);
    RAM64(in=in, load=s3, address=address[3..8], out=t3);
    RAM64(in=in, load=s4, address=address[3..8], out=t4);
    RAM64(in=in, load=s5, address=address[3..8], out=t5);
    RAM64(in=in, load=s6, address=address[3..8], out=t6);
    RAM64(in=in, load=s7, address=address[3..8], out=t7);
    RAM64(in=in, load=s8, address=address[3..8], out=t8);
    Mux8Way16(a=t1, b=t2, c=t3, d=t4, e=t5, f=t6, g=t7, h=t8,
sel=address[0..2], out=out);
}
```

RAM512 TEST:



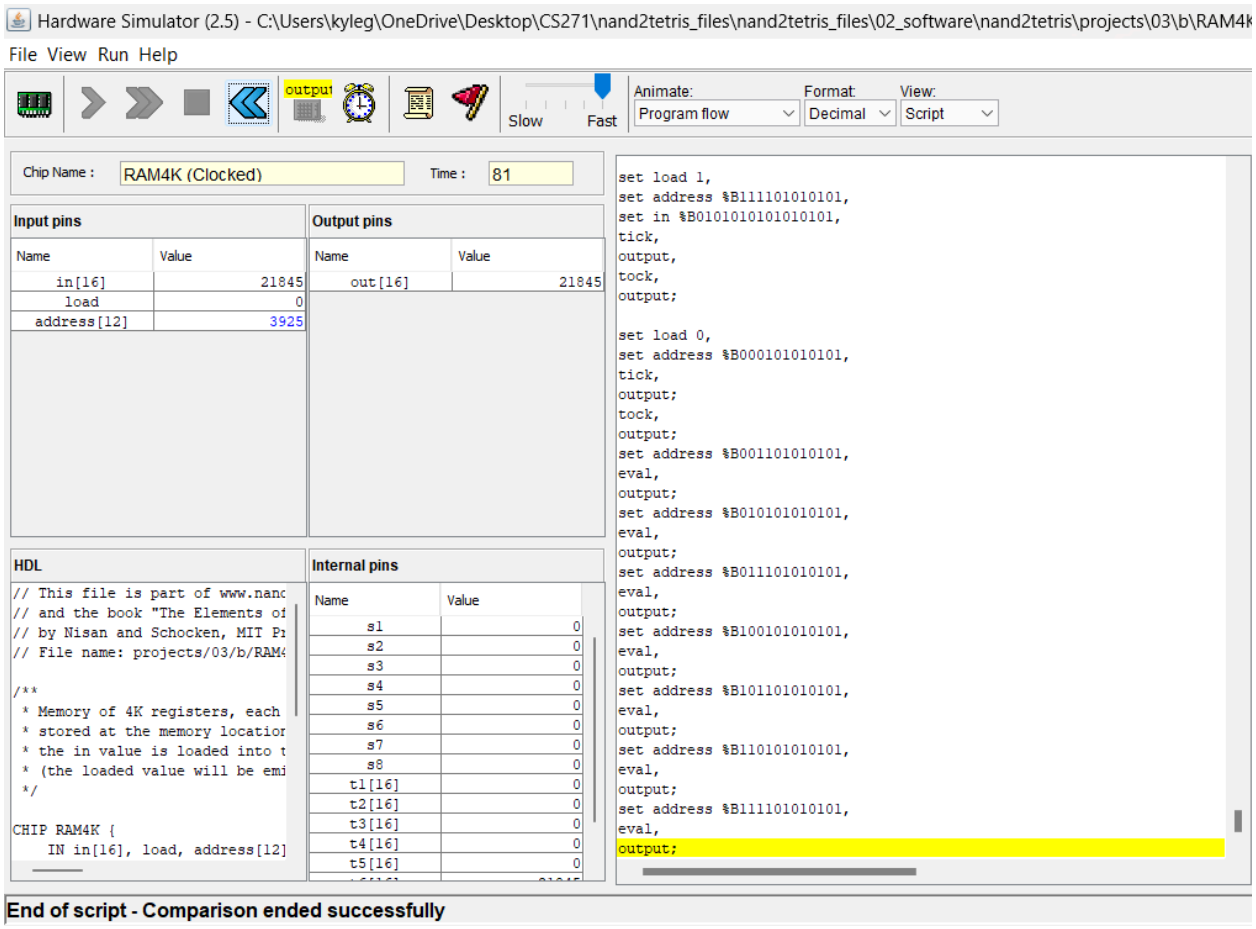
RAM4K CHIP:

```
CHIP RAM4K {
```

```
    IN in[16], load, address[12];
    OUT out[16];

    PARTS:
    // Putting 8 512-bit RAM together
    DMux8Way(in=load, sel=address[0..2], a=s1, b=s2, c=s3, d=s4, e=s5,
f=s6, g=s7, h=s8);
    RAM512(in=in, load=s1, address=address[3..11], out=t1);
    RAM512(in=in, load=s2, address=address[3..11], out=t2);
    RAM512(in=in, load=s3, address=address[3..11], out=t3);
    RAM512(in=in, load=s4, address=address[3..11], out=t4);
    RAM512(in=in, load=s5, address=address[3..11], out=t5);
    RAM512(in=in, load=s6, address=address[3..11], out=t6);
    RAM512(in=in, load=s7, address=address[3..11], out=t7);
    RAM512(in=in, load=s8, address=address[3..11], out=t8);
    Mux8Way16(a=t1, b=t2, c=t3, d=t4, e=t5, f=t6, g=t7, h=t8,
sel=address[0..2], out=out);
}
```

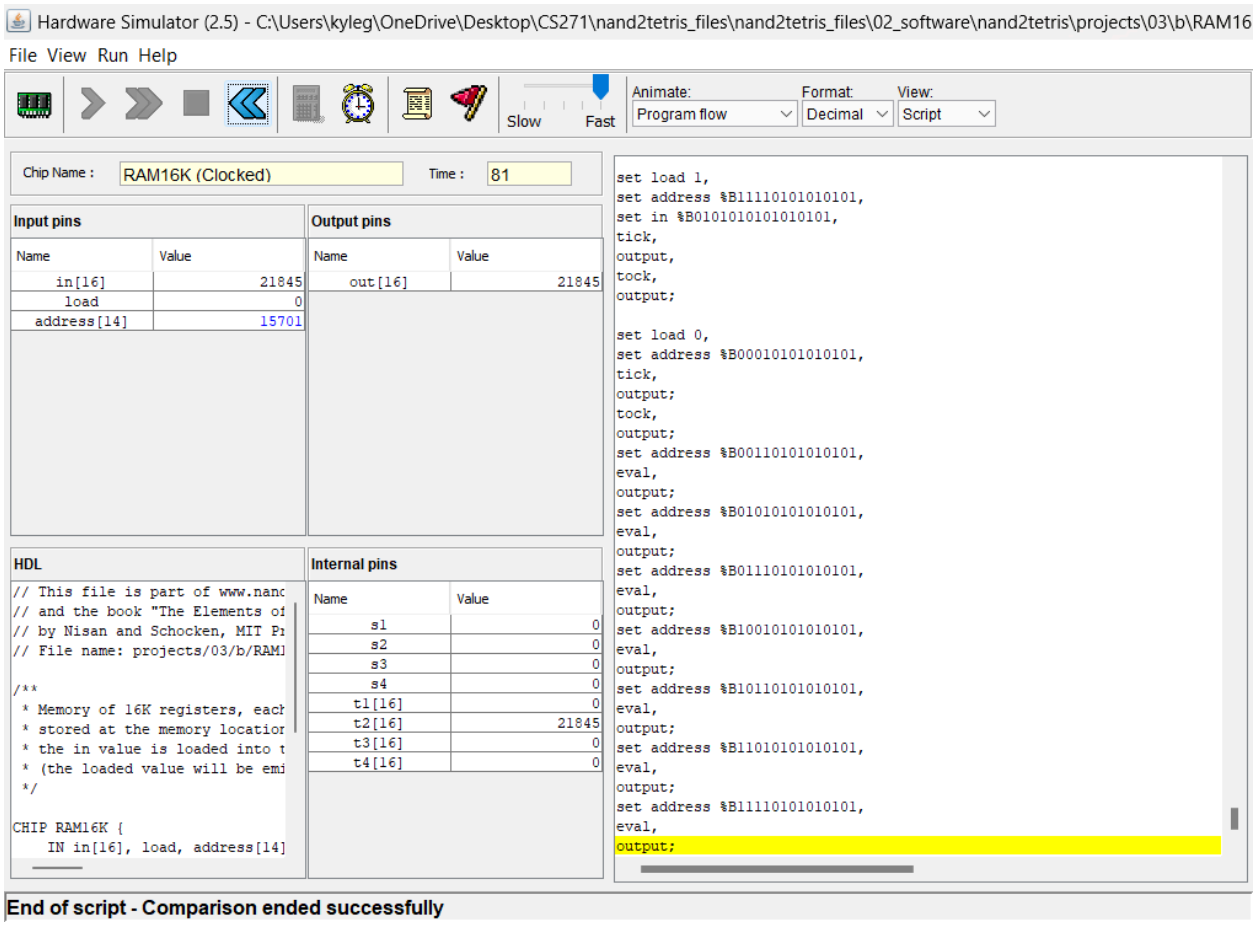
RAM4K TEST:



RAM16K CHIP:

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

    PARTS:
    // Putting 4 4K-RAM together
    DMux4Way(in=load, sel=address[0..1], a=s1, b=s2, c=s3, d=s4);
    RAM4K(in=in, load=s1, address=address[2..13], out=t1);
    RAM4K(in=in, load=s2, address=address[2..13], out=t2);
    RAM4K(in=in, load=s3, address=address[2..13], out=t3);
    RAM4K(in=in, load=s4, address=address[2..13], out=t4);
    Mux4Way16(a=t1, b=t2, c=t3, d=t4, sel=address[0..1], out=out);
}
```



PC CHIP:

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

    PARTS:
    // I had to look up assistance for this chip
    // computes increment of input
    Inc16(in=loop, out=incre);
    // gate for t input with increment
    Mux16(a=loop, b=incre, sel=inc, out=s1);
    // gate for recent decision and t-1 input
    Mux16(a=s1, b=in, sel=load, out=s2);
    //gate for resetting the clock or not
    Mux16(a=s2, b[0..15]=false, sel=reset, out=s3);
    // storing result no matter what, along with assigning a variable for
the loop and output
```

```
    Register(in=s3, load=true, out=out, out=loop);
}
```

PC TEST: