Contents

1 README

```
roigreenberg,inbaravni
==============================================================================
Roi Greenberg, ID 30557123, roi.greenberg@mail.huji.ac.il
Inbar Avni, ID 201131760, inbar.avni@mail.huji.ac.il
==============================================================================

                           Project 3 - Sequential Chips
                           ----------------------------


Submitted Files
---------------
README - This file.
Bit.hdl - 1-bit register.
Register.hdl - 16-bit register.
RAM8.hdl - 16-bit / 8-register memory.
RAM64.hdl - 16-bit / 64-register memory.
RAM512.hdl - 16-bit / 512-register memory.
RAM4K.hdl - 16-bit / 4096-register memory.
RAM16K.hdl - 16-bit / 16384-register memory.
PC.hdl - 16-bit program counter.


Remarks
-------
:)
```

2 a/Bit.hdl

```
// 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:
    // save the state or have a new value, depends on the load value
    Mux(a=out1, b=in, sel=load, out=in1);
    // set the value in case of maintaining the value
    DFF(in=in1, out=out1, out=out);
}
```

3 a/PC.hdl

```
1  // This file is part of www.nand2tetris.org
2  // and the book "The Elements of Computing Systems"
3  // by Nisan and Schocken, MIT Press.
4  // File name: projects/03/a/PC.hdl
5
6  /**
7   * A 16-bit counter with load and reset control bits.
8   * if      (reset[t] == 1) out[t+1] = 0
9   * else if (load[t] == 1)  out[t+1] = in[t]
10  * else if (inc[t] == 1)   out[t+1] = out[t] + 1  (integer addition)
11  * else                    out[t+1] = out[t]
12  */
13
14 CHIP PC {
15     IN in[16],load,inc,reset;
16     OUT out[16];
17
18     PARTS:
19     // check if out is a new value, and not maintaining the out value
20     Or(a=load, b=inc, out=tmp);
21     Or(a=tmp, b=reset, out=superload);
22     // set the out value in case of increment
23     Inc16(in=out1, out=afterinc);
24     // determine the value for out
25     Mux16(a=afterload, b=false, sel=reset, out=afterreset);
26     Mux16(a=afterinc, b=in, sel=load, out=afterload);
27     // registering
28     Register(in=afterreset, load=superload, out=out1, out=out);
29 }
```

4 a/RAM64.hdl

```
// 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:
    // set the value specified by in to certain ram8 according the address
    DMux8Way(in=load, sel=address[3..5], a=load1, b=load2, c=load3, d=load4, e=load5, f=load6, g=load7, h=load8);
    // do ram8 to all the eight
    RAM8(in=in, out=out1, load=load1, address=address[0..2]);
    RAM8(in=in, out=out2, load=load2, address=address[0..2]);
    RAM8(in=in, out=out3, load=load3, address=address[0..2]);
    RAM8(in=in, out=out4, load=load4, address=address[0..2]);
    RAM8(in=in, out=out5, load=load5, address=address[0..2]);
    RAM8(in=in, out=out6, load=load6, address=address[0..2]);
    RAM8(in=in, out=out7, load=load7, address=address[0..2]);
    RAM8(in=in, out=out8, load=load8, address=address[0..2]);
    // output the selected register value
    Mux8Way16(a=out1, b=out2, c=out3, d=out4, e=out5, f=out6, g=out7, h=out8, sel=address[3..5], out=out);
}
```

5 a/RAM8.hdl

```
// 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:
    // set the value specified by in to certain register according the address
    DMux8Way(in=load, sel=address, a=load1, b=load2, c=load3, d=load4, e=load5, f=load6, g=load7, h=load8);
    // do registers to all the eight
    Register(in=in, out=reg1, load=load1);
    Register(in=in, out=reg2, load=load2);
    Register(in=in, out=reg3, load=load3);
    Register(in=in, out=reg4, load=load4);
    Register(in=in, out=reg5, load=load5);
    Register(in=in, out=reg6, load=load6);
    Register(in=in, out=reg7, load=load7);
    Register(in=in, out=reg8, load=load8);
    // output the selected register value
    Mux8Way16(a=reg1, b=reg2, c=reg3, d=reg4, e=reg5, f=reg6, g=reg7, h=reg8, sel=address, out=out);
}
```

6 a/Register.hdl

```
// 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:
    // 16 times of bit chip
    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]);
}
```

7 b/RAM16K.hdl

```
// 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:
    // set the value specified by in to certain ram4k according the address
    DMux4Way(in=load, sel=address[12..13], a=load1, b=load2, c=load3, d=load4);
    // do ram4k to all the four
    RAM4K(in=in, out=out1, load=load1, address=address[0..11]);
    RAM4K(in=in, out=out2, load=load2, address=address[0..11]);
    RAM4K(in=in, out=out3, load=load3, address=address[0..11]);
    RAM4K(in=in, out=out4, load=load4, address=address[0..11]);
    // output the selected register value
    Mux4Way16(a=out1, b=out2, c=out3, d=out4, sel=address[12..13], out=out);
}
```

8 b/RAM4K.hdl

```
// 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:
    // set the value specified by in to certain ram512 according the address
    DMux8Way(in=load, sel=address[9..11], a=load1, b=load2, c=load3, d=load4, e=load5, f=load6, g=load7, h=load8);
    // do ram512 to all the eight
    RAM512(in=in, out=out1, load=load1, address=address[0..8]);
    RAM512(in=in, out=out2, load=load2, address=address[0..8]);
    RAM512(in=in, out=out3, load=load3, address=address[0..8]);
    RAM512(in=in, out=out4, load=load4, address=address[0..8]);
    RAM512(in=in, out=out5, load=load5, address=address[0..8]);
    RAM512(in=in, out=out6, load=load6, address=address[0..8]);
    RAM512(in=in, out=out7, load=load7, address=address[0..8]);
    RAM512(in=in, out=out8, load=load8, address=address[0..8]);
    // output the selected register value
    Mux8Way16(a=out1, b=out2, c=out3, d=out4, e=out5, f=out6, g=out7, h=out8, sel=address[9..11], out=out);
}
```

9 b/RAM512.hdl

```
// 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:
    // set the value specified by in to certain ram64 according the address
    DMux8Way(in=load, sel=address[6..8], a=load1, b=load2, c=load3, d=load4, e=load5, f=load6, g=load7, h=load8);
    // do ram64 to all the eight
    RAM64(in=in, out=out1, load=load1, address=address[0..5]);
    RAM64(in=in, out=out2, load=load2, address=address[0..5]);
    RAM64(in=in, out=out3, load=load3, address=address[0..5]);
    RAM64(in=in, out=out4, load=load4, address=address[0..5]);
    RAM64(in=in, out=out5, load=load5, address=address[0..5]);
    RAM64(in=in, out=out6, load=load6, address=address[0..5]);
    RAM64(in=in, out=out7, load=load7, address=address[0..5]);
    RAM64(in=in, out=out8, load=load8, address=address[0..5]);
    // output the selected register value
    Mux8Way16(a=out1, b=out2, c=out3, d=out4, e=out5, f=out6, g=out7, h=out8, sel=address[6..8], out=out);
}
```