Contents

1 Basic Test Results

```
********** TESTING FOLDER STRUCTURE START **********
Checking your submission for presence of invalid (non-ASCII) characters...
No invalid characters found.
Submission logins are: linorcohen
Is this OK?
**********  TESTING FOLDER STRUCTURE END  **********

********** PROJECT TEST START **********
Testing.
a/Bit passed test.
a/PC passed test.
a/RAM64 passed test.
a/RAM8 passed test.
a/Register passed test.
b/RAM16K passed test.
b/RAM4K passed test.
b/RAM512 passed test.
**********  PROJECT TEST END  **********

Note: the tests you see above are all the presubmission tests
for this project. The tests might not check all the different
parts of the project or all corner cases, so write your own
tests and use them!
```

2 AUTHORS

```
1 linorcohen
2 Partner 1: Linor Cohen, linor.cohen@mail.huji.ac.il, 318861226
3 Remarks:
```

3 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:
    Mux(a=w2, b=in, sel=load, out=w1);
    DFF(in=w1, out=w2, out=out);
}
```

4 a/PC.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/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=out1, out=w1);
    Mux16(a=out1, b=w1, sel=inc, out=w2);
    Mux16(a=w2, b=in, sel=load, out=w3);
    Mux16(a=w3, b=false, sel=reset, out=w4);
    Register(in=w4, load=true, out=out1, out=out);
}
```

5 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:
    DMux8Way(in=load, sel=address[0..2], a=r1, b=r2, c=r3, d=r4, e=r5, f=r6, g=r7, h=r8);
    RAM8(in=in, load=r1, address=address[3..5], out=w1);
    RAM8(in=in, load=r2, address=address[3..5], out=w2);
    RAM8(in=in, load=r3, address=address[3..5], out=w3);
    RAM8(in=in, load=r4, address=address[3..5], out=w4);
    RAM8(in=in, load=r5, address=address[3..5], out=w5);
    RAM8(in=in, load=r6, address=address[3..5], out=w6);
    RAM8(in=in, load=r7, address=address[3..5], out=w7);
    RAM8(in=in, load=r8, address=address[3..5], out=w8);
    Mux8Way16(a=w1, b=w2, c=w3, d=w4, e=w5, f=w6, g=w7, h=w8, sel=address[0..2], out=out);
}
```

6 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:
    DMux8Way(in=load, sel=address, a=r1, b=r2, c=r3, d=r4, e=r5, f=r6, g=r7, h=r8);
    Register(in=in, load=r1, out=w1);
    Register(in=in, load=r2, out=w2);
    Register(in=in, load=r3, out=w3);
    Register(in=in, load=r4, out=w4);
    Register(in=in, load=r5, out=w5);
    Register(in=in, load=r6, out=w6);
    Register(in=in, load=r7, out=w7);
    Register(in=in, load=r8, out=w8);
    Mux8Way16(a=w1, b=w2, c=w3, d=w4, e=w5, f=w6, g=w7, h=w8, sel=address, out=out);
}
```

7 a/Register.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/Register.hdl
5
6  /**
7   * 16-bit register:
8   * If load[t] == 1 then out[t+1] = in[t]
9   * else out does not change
10  */
11
12 CHIP Register {
13     IN in[16], load;
14     OUT out[16];
15
16     PARTS:
17     Bit(in=in[0], load=load, out=out[0]);
18     Bit(in=in[1], load=load, out=out[1]);
19     Bit(in=in[2], load=load, out=out[2]);
20     Bit(in=in[3], load=load, out=out[3]);
21     Bit(in=in[4], load=load, out=out[4]);
22     Bit(in=in[5], load=load, out=out[5]);
23     Bit(in=in[6], load=load, out=out[6]);
24     Bit(in=in[7], load=load, out=out[7]);
25     Bit(in=in[8], load=load, out=out[8]);
26     Bit(in=in[9], load=load, out=out[9]);
27     Bit(in=in[10], load=load, out=out[10]);
28     Bit(in=in[11], load=load, out=out[11]);
29     Bit(in=in[12], load=load, out=out[12]);
30     Bit(in=in[13], load=load, out=out[13]);
31     Bit(in=in[14], load=load, out=out[14]);
32     Bit(in=in[15], load=load, out=out[15]);
33 }
```

8 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:
    DMux4Way(in=load, sel=address[0..1], a=r1, b=r2, c=r3, d=r4);
    RAM4K(in=in, load=r1, address=address[2..13], out=w1);
    RAM4K(in=in, load=r2, address=address[2..13], out=w2);
    RAM4K(in=in, load=r3, address=address[2..13], out=w3);
    RAM4K(in=in, load=r4, address=address[2..13], out=w4);
    Mux4Way16(a=w1, b=w2, c=w3, d=w4, sel=address[0..1], out=out);
}
```

9 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:
    DMux8Way(in=load, sel=address[0..2], a=r1, b=r2, c=r3, d=r4, e=r5, f=r6, g=r7, h=r8);
    RAM512(in=in, load=r1, address=address[3..11], out=w1);
    RAM512(in=in, load=r2, address=address[3..11], out=w2);
    RAM512(in=in, load=r3, address=address[3..11], out=w3);
    RAM512(in=in, load=r4, address=address[3..11], out=w4);
    RAM512(in=in, load=r5, address=address[3..11], out=w5);
    RAM512(in=in, load=r6, address=address[3..11], out=w6);
    RAM512(in=in, load=r7, address=address[3..11], out=w7);
    RAM512(in=in, load=r8, address=address[3..11], out=w8);
    Mux8Way16(a=w1, b=w2, c=w3, d=w4, e=w5, f=w6, g=w7, h=w8, sel=address[0..2], out=out);
}
```

10 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:
    DMux8Way(in=load, sel=address[0..2], a=r1, b=r2, c=r3, d=r4, e=r5, f=r6, g=r7, h=r8);
    RAM64(in=in, load=r1, address=address[3..8], out=w1);
    RAM64(in=in, load=r2, address=address[3..8], out=w2);
    RAM64(in=in, load=r3, address=address[3..8], out=w3);
    RAM64(in=in, load=r4, address=address[3..8], out=w4);
    RAM64(in=in, load=r5, address=address[3..8], out=w5);
    RAM64(in=in, load=r6, address=address[3..8], out=w6);
    RAM64(in=in, load=r7, address=address[3..8], out=w7);
    RAM64(in=in, load=r8, address=address[3..8], out=w8);
    Mux8Way16(a=w1, b=w2, c=w3, d=w4, e=w5, f=w6, g=w7, h=w8, sel=address[0..2], out=out);
}
```