Contents

1 Basic Test Results

```
********** TESTING FOLDER STRUCTURE START **********
Running test3.sh:
Your logins are: muaz.abdeen, is that ok?

********** TESTING FOLDER STRUCTURE END **********

********** PROJECT TEST START **********

a/Bit passed test
a/Register passed test

********** PROJECT TEST END **********
```

2 README

```
muaz.abdeen
================================================================================
Muaz Abdeen, ID 300575297, muaz.abdeen@mail.huji.ac.il
================================================================================

                           Project 3 - Sequential Logic
                           ----------------------------


Submitted Files
---------------
(1)  README             - This file.
(2)  Bit.hdl            - The 1-bit register implementation.
(3)  Register.hdl        - The 16-bit register implementation.
(4)  RAM8.hdl            - The 8-register RAM implementation.
(5)  RAM64.hdl            - The RAM64 implementation.
(6)  RAM512.hdl            - The RAM512 implementation.
(7)  RAM4K.hdl            - The RAM4K implementation.
(8)  RAM16K.hdl            - The RAM16K implementation.
(9)  PC.hdl            - The implementation of the counter.


Remarks
-------
* ...
```

3 ./a/Bit.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/Bit.hdl
5
6  /**
7   * 1-bit register:
8   * If load[t] == 1 then out[t+1] = in[t]
9   *                 else out does not change (out[t+1] = out[t])
10  */
11
12 CHIP Bit {
13     IN in, load;
14     OUT out;
15
16     PARTS:
17     // Determine if to load the input or keep the current data
18     Mux(a=current ,b=in ,sel=load ,out=res);
19
20     // Using the DFF to make a time delay
21     DFF(in=res ,out=current, out=out);
22 }
```

4 ./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     // Increment the current data
20     Inc16(in=current ,out=incRes);
21
22     // Determine if to increment the counter
23     Mux16(a=current ,b=incRes ,sel=inc ,out=res0);
24
25     // Determine if to load the input
26     Mux16(a=res0 ,b=in ,sel=load ,out=res1);
27
28     // Determine if to reset the counter
29     Mux16(a=res1 ,b=false ,sel=reset ,out=res2);
30
31     // Save the final status
32     Register(in=res2 ,load=true ,out=current ,out=out);
33
34 }
```

5 ./a/RAM64.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/RAM64.hdl
5
6  /**
7   * Memory of 64 registers, each 16 bit-wide. Out holds the value
8   * stored at the memory location specified by address. If load==1, then
9   * the in value is loaded into the memory location specified by address
10  * (the loaded value will be emitted to out from the next time step onward).
11  */
12
13 CHIP RAM64 {
14     IN in[16], load, address[6];
15     OUT out[16];
16
17     PARTS:
18     // determine the RAM8 chip to write to
19     DMux8Way(in=load ,sel=address[3..5] ,a=ram0 ,b=ram1 ,c=ram2 ,d=ram3 ,e=ram4 ,f=ram5 ,g=ram6 ,h=ram7);
20
21     // determine the inner Register chip to write to and read from
22     RAM8(in=in ,load=ram0 ,address=address[0..2] ,out=out0);
23     RAM8(in=in ,load=ram1 ,address=address[0..2] ,out=out1);
24     RAM8(in=in ,load=ram2 ,address=address[0..2] ,out=out2);
25     RAM8(in=in ,load=ram3 ,address=address[0..2] ,out=out3);
26     RAM8(in=in ,load=ram4 ,address=address[0..2] ,out=out4);
27     RAM8(in=in ,load=ram5 ,address=address[0..2] ,out=out5);
28     RAM8(in=in ,load=ram6 ,address=address[0..2] ,out=out6);
29     RAM8(in=in ,load=ram7 ,address=address[0..2] ,out=out7);
30
31     // determine the RAM8 chip to read from
32     Mux8Way16(a=out0 ,b=out1 ,c=out2 ,d=out3 ,e=out4 ,f=out5 ,g=out6 ,h=out7 ,sel=address[3..5] ,out=out);
33 }
```

6 ./a/RAM8.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/RAM8.hdl
5
6  /**
7   * Memory of 8 registers, each 16 bit-wide. Out holds the value
8   * stored at the memory location specified by address. If load==1, then
9   * the in value is loaded into the memory location specified by address
10  * (the loaded value will be emitted to out from the next time step onward).
11  */
12
13 CHIP RAM8 {
14     IN in[16], load, address[3];
15     OUT out[16];
16
17     PARTS:
18     // determine the Register chip to write to
19     DMux8Way(in=load ,sel=address ,a=reg0 ,b=reg1 ,c=reg2 ,d=reg3 ,e=reg4 ,f=reg5 ,g=reg6 ,h=reg7);
20
21     // determine the Register chip to write to and read from
22     Register(in=in ,load=reg0 ,out=out0);
23     Register(in=in ,load=reg1 ,out=out1);
24     Register(in=in ,load=reg2 ,out=out2);
25     Register(in=in ,load=reg3 ,out=out3);
26     Register(in=in ,load=reg4 ,out=out4);
27     Register(in=in ,load=reg5 ,out=out5);
28     Register(in=in ,load=reg6 ,out=out6);
29     Register(in=in ,load=reg7 ,out=out7);
30
31     // determine the Register chip to read from
32     Mux8Way16(a=out0 ,b=out1 ,c=out2 ,d=out3 ,e=out4 ,f=out5 ,g=out6 ,h=out7 ,sel=address ,out=out);
33 }
```

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     // Using the 1-bit register on every bit of the input pin
18     Bit(in=in[0] ,load=load ,out=out[0]);
19     Bit(in=in[1] ,load=load ,out=out[1]);
20     Bit(in=in[2] ,load=load ,out=out[2]);
21     Bit(in=in[3] ,load=load ,out=out[3]);
22     Bit(in=in[4] ,load=load ,out=out[4]);
23     Bit(in=in[5] ,load=load ,out=out[5]);
24     Bit(in=in[6] ,load=load ,out=out[6]);
25     Bit(in=in[7] ,load=load ,out=out[7]);
26     Bit(in=in[8] ,load=load ,out=out[8]);
27     Bit(in=in[9] ,load=load ,out=out[9]);
28     Bit(in=in[10] ,load=load ,out=out[10]);
29     Bit(in=in[11] ,load=load ,out=out[11]);
30     Bit(in=in[12] ,load=load ,out=out[12]);
31     Bit(in=in[13] ,load=load ,out=out[13]);
32     Bit(in=in[14] ,load=load ,out=out[14]);
33     Bit(in=in[15] ,load=load ,out=out[15]);
34 }
```

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:
    // determine the RAM4K chip to write to
    DMux4Way(in=load ,sel=address[12..13] ,a=ram0 ,b=ram1 ,c=ram2 ,d=ram3);

    // Recursivley determine the inner Register chip to write to and read from
    RAM4K(in=in ,load=ram0 ,address=address[0..11] ,out=out0);
    RAM4K(in=in ,load=ram1 ,address=address[0..11] ,out=out1);
    RAM4K(in=in ,load=ram2 ,address=address[0..11] ,out=out2);
    RAM4K(in=in ,load=ram3 ,address=address[0..11] ,out=out3);

    // determine the RAM4K chip to read from
    Mux4Way16(a=out0 ,b=out1 ,c=out2 ,d=out3 ,sel=address[12..13] ,out=out);
}
```

9 ./b/RAM4K.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/b/RAM4K.hdl
5
6  /**
7   * Memory of 4K registers, each 16 bit-wide. Out holds the value
8   * stored at the memory location specified by address. If load==1, then
9   * the in value is loaded into the memory location specified by address
10  * (the loaded value will be emitted to out from the next time step onward).
11  */
12
13 CHIP RAM4K {
14     IN in[16], load, address[12];
15     OUT out[16];
16
17     PARTS:
18     // determine the RAM512 chip to write to
19     DMux8Way(in=load ,sel=address[9..11] ,a=ram0 ,b=ram1 ,c=ram2 ,d=ram3 ,e=ram4 ,f=ram5 ,g=ram6 ,h=ram7);
20
21     // Recursivley determine the inner Register chip to write to and read from
22     RAM512(in=in ,load=ram0 ,address=address[0..8] ,out=out0);
23     RAM512(in=in ,load=ram1 ,address=address[0..8] ,out=out1);
24     RAM512(in=in ,load=ram2 ,address=address[0..8] ,out=out2);
25     RAM512(in=in ,load=ram3 ,address=address[0..8] ,out=out3);
26     RAM512(in=in ,load=ram4 ,address=address[0..8] ,out=out4);
27     RAM512(in=in ,load=ram5 ,address=address[0..8] ,out=out5);
28     RAM512(in=in ,load=ram6 ,address=address[0..8] ,out=out6);
29     RAM512(in=in ,load=ram7 ,address=address[0..8] ,out=out7);
30
31     // determine the RAM512 chip to read from
32     Mux8Way16(a=out0 ,b=out1 ,c=out2 ,d=out3 ,e=out4 ,f=out5 ,g=out6 ,h=out7 ,sel=address[9..11] ,out=out);
33 }
```

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:
    // determine the RAM64 chip to write to
    DMux8Way(in=load ,sel=address[6..8] ,a=ram0 ,b=ram1 ,c=ram2 ,d=ram3 ,e=ram4 ,f=ram5 ,g=ram6 ,h=ram7);

    // Recursivley determine the inner Register chip to write to and read from
    RAM64(in=in ,load=ram0 ,address=address[0..5] ,out=out0);
    RAM64(in=in ,load=ram1 ,address=address[0..5] ,out=out1);
    RAM64(in=in ,load=ram2 ,address=address[0..5] ,out=out2);
    RAM64(in=in ,load=ram3 ,address=address[0..5] ,out=out3);
    RAM64(in=in ,load=ram4 ,address=address[0..5] ,out=out4);
    RAM64(in=in ,load=ram5 ,address=address[0..5] ,out=out5);
    RAM64(in=in ,load=ram6 ,address=address[0..5] ,out=out6);
    RAM64(in=in ,load=ram7 ,address=address[0..5] ,out=out7);

    // determine the RAM64 chip to read from
    Mux8Way16(a=out0 ,b=out1 ,c=out2 ,d=out3 ,e=out4 ,f=out5 ,g=out6 ,h=out7 ,sel=address[6..8] ,out=out);
}
```