

Team members:

A: Matthew Kovar

B: Sung Ki Ling

C: Christopher O'Toole

D: Yunchao Zhang

The work was distributed according to the following pattern:

Chip Name	Implementer	Tester	Documentation Reviewer
Bit.hdl	Matthew Kovar	Sung Ki Ling	Yunchao Zhang
Register.hdl	Sung Ki Ling	Christopher O'Toole	Matthew Kovar
RAM8.hdl	Christopher O'Toole	Yunchao Zhang	Sung Ki Ling
RAM64.hdl	Yunchao Zhang	Matthew Kovar	Christopher O'Toole
RAM512.hdl	Matthew Kovar	Sung Ki Ling	Yunchao Zhang
RAM4K.hdl	Sung Ki Ling	Christopher O'Toole	Matthew Kovar
RAM16K.hdl	Christopher O'Toole	Yunchao Zhang	Sung Ki Ling
PC.hdl	Yunchao Zhang	Matthew Kovar	Christopher O'Toole

The chips were tested by each tester using the default tests that came with the Nand2Tetris project files. Screenshots of successful test runs are included at the end of this document.

All .hdl files are included in the containing .zip file.

**Detailed implementation documentation is included in all .hdl files as comments (lines preceded by // or /\*).**

A general overview of the function of each chip is provided below. This brief documentation is provided in addition to the detailed implementation documentation contained as comments directly within every .hdl file.

The Bit chip is a 1-bit register that is used to store a single bit in memory. It was implemented using a mux using the load input as the selector bit to decide if a bit should be written to the register. The D-flip-flop is then used to store the bit and make it accessible at a later time. The Bit chip forms the basis for all future registers and RAM units.

The Register chip is just sixteen Bit units tied together, one after the other, used to hold 16-bit words instead of just 1-bit words. It functions in the same way as the Bit chip, except now the input is 16 bits long.

The RAM8 chip is just eight Register units tied together, one after the other, used to hold eight distinct 16-bit words. Since only one register is written to at any given time, a 3-bit address is provided, where the decimal conversion corresponds to which of the eight registers is being referred to. The Dmux is used to interpret the 3-bit address to select the correct register. Finally, a mux is used to return the correct value held by the register at the given address.

The RAM64 chip is just eight RAM8 units tied together, but now with a 6-bit address provided, since  $2^6 = 64$ . This chip functions identically to the RAM8 unit, except now up to sixty-four individual 16-bit words can be stored instead of only eight. The only other distinction between this unit and the RAM8 unit is that the first three bits of the provided address are used to select which RAM8 unit to access while the remaining bits are used to select the correct register within the RAM8 unit.

The RAM512 chip is just eight RAM64 units tied together, but now with a 9-bit address provided, since  $2^9 = 512$ . This chip functions identically to the RAM64 unit, except now up to 512 individual 16-bit words can be stored instead of only 64.

The RAM4K chip is just eight RAM512 units tied together, but now with a 12-bit address provided, since  $2^{12} = 4096$ . This chip functions identically to the RAM512 unit, except now up to 4096 individual 16-bit words can be stored instead of only 512.

The RAM16K chip is just four RAM4K units tied together, but now with a 14-bit address provided, since  $2^{14} = 16384$ . This chip functions identically to the RAM4K unit, except now up to 16384 individual 16-bit words can be stored instead of only 4096.

The program counter is a register that contains the address (location) of the instruction being executed at the current time. As each instruction, the program counter increases its stored value by 1. After each instruction is fetched, the program counter points to the next instruction in the sequence. As such, it is implemented using the Inc16 chip to increase the value. It uses three additional Mux16 chips to correspond to each of the three provided flags: load, inc, and reset. Based on the mux selection, the appropriate action is taken in the register.

Screenshots of successful tests are included below:

## CS 2200 HW 5 (Nand2Tetris 03)

Bit.hdl test:

The screenshot displays the Hardware Simulator (2.5) window. The title bar shows the file path: D:\Projects\School\CS 2200\HW5\hdl\CS2200-Nand2Tetris03\a\Bit.hdl.

**File View Run Help**

**Simulation Controls:** Includes icons for logic gates, a clock, a document, and a speaker. A slider controls the speed from "Slow" to "Fast". Buttons for "Animate:", "Format:", and "View:" are present, with dropdown menus for "Program flow", "Decimal", and "Script".

**Chip Name:** Bit (Clocked)      **Time:** 107

Input pins		Output pins	
Name	Value	Name	Value
in	1	out	0
load	0		

  

Internal pins	
Name	Value
feedback	0
dffin	0

  

**HDL Code:**

```
/**
 * Author:Matthew Kovar
 * Date:2018/11/07
 * 1-bit register.
 * If load[t]=1 then out[t+1] =
 * else out does not change
 */

// Documentation is included as
// The Bit chip returns the input value if load is 1,
// otherwise if load is 0, it returns the previous output value
// it held previously
```

**Script Output:**

```
output;
tick,
output;

set in 1,
set load 0,
tick,
output;

tick,
output;

set in 1,
set load 0,
tick,
output;

tick,
output;

set in 1,
set load 0,
tick,
output;

tick,
output;
```

**End of script - Comparison ended successfully**

## Register.hdl test:

Hardware Simulator (2.5) - D:\Projects\School\CS 2200\HW5\hdl\CS2200-Nand2Tetris03\Register.hdl

File View Run Help

Animate: Program flow Format: Decimal View: Script

Chip Name: Register (Clocked) Time: 74

Input pins		Output pins	
Name	Value	Name	Value
in[16]	32767	out[16]	32767
load	1		

Internal pins	
Name	Value

HDL

```

/**
 * Author: Sung Ki Ling
 * Date: 2018/11/08
 * 16-bit register:
 * If load[t] == 1 then out[t+1]
 * else out does not change
 */

// Let $b_i$ for $i$ \in [0, 15]
// Two distinct cases describe t
// - Case 1 (load = 0): $b_i = i$
// - Case 2 (load = 1): $b_i = i$
// These cases follow from the c
// The fact that this holds for

```

```

set load 1,
tick,
output;

tock,
output;

set in $B1011111111111111,
set load 0,
tick,
output;

tock,
output;

set load 1,
tick,
output;

tock,
output;

set in $B0111111111111111,
set load 0,
tick,
output;

tock,
output;

set load 1,
tick,
output;

tock,
output;

```

End of script - Comparison ended successfully

RAM8.hdl test:

Hardware Simulator (2.5) - D:\Projects\School\CS 2200\HW5\hdl\CS2200-Nand2Tetris03\RAM8.hdl

File View Run Help

Chip Name : RAM8 (Clocked) Time : 46

Input pins		Output pins	
Name	Value	Name	Value
in[16]	21845	out[16]	21845
load	0		
address[3]	7		

Internal pins	
Name	Value
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
o1[16]	21845
o2[16]	21845
o3[16]	21845
o4[16]	21845
o5[16]	21845
o6[16]	21845

HDL

```

/**
 * Author: Christopher O'Toole
 * Date: 2018/11/08
 * Memory of 8 registers, each 16 bits
 * stored at the memory location
 * the in value is loaded into t
 * (the loaded value will be emi
 */
// The RAM8 chip is composed of
// the specified 16-bit value in
// that address will keep its c
// For example, if address==000
// register to the value held in

```

```

set load 1,
set address 7,
set in $B0101010101010101,
tick,
output,
tock,
output;

set load 0,
set address 0,
tick,
output;
tock,
output;
set address 1,
eval,
output;
set address 2,
eval,
output;
set address 3,
eval,
output;
set address 4,
eval,
output;
set address 5,
eval,
output;
set address 6,
eval,
output;
set address 7,
eval,
output;

```

End of script - Comparison ended successfully

RAM64.hdl test:

Hardware Simulator (2.5) - D:\Projects\School\CS 2200\HW5\hds\CS2200-Nand2Tetris03\RAM64.hdl

File View Run Help

Chip Name: RAM64 (Clocked) Time: 81

Input pins		Output pins	
Name	Value	Name	Value
in[16]	21845	out[16]	21845
load	0		
address[6]	61		

HDL

```

/**
 * Author: Yunchao Zhang
 * Date: 2018-11-09
 * Memory of 64 registers, each
 * stored at the memory location
 * the in value is loaded into t
 * (the loaded value will be emi
 */
// The RAM64 chip is just 64 16-
// We load in the same way as wi
// loading 16-bit words instead
// an address that tells which c

```

Internal pins	
Name	Value
11	0
out1[16]	0
12	0
out2[16]	0
13	0
out3[16]	0
14	0
out4[16]	0
15	0
out5[16]	0
16	0
out6[16]	21845
17	0
out7[16]	0

```

set load 1,
set address $B111101,
set in $B0101010101010101,
tick,
output,
tock,
output;

set load 0,
set address $B000101,
tick,
output;
tock,
output;
set address $B001101,
eval,
output;
set address $B010101,
eval,
output;
set address $B011101,
eval,
output;
set address $B100101,
eval,
output;
set address $B101101,
eval,
output;
set address $B110101,
eval,
output;
set address $B111101,
eval,
output;

```

End of script - Comparison ended successfully

# CS 2200 HW 5 (Nand2Tetris 03)

RAM512.hdl test:

Hardware Simulator (2.5) - D:\Projects\School\CS 2200\HW5\hdl\CS2200-Nand2Tetris03\b\RAM512.hdl

File View Run Help

Chip Name : RAM512 (Clocked) Time : 81

Input pins

Name	Value
in[16]	21845
load	0
address[9]	490

Output pins

Name	Value
out[16]	21845

HDL

```

/**
 * Author:Matthew Kovar
 * Date:2018/11/07
 * Memory of 512 registers, each
 * stored at the memory location
 * the 'in' value is loaded into
 * (the loaded value will be given)
 */
// Documentation is included as
// The RAM512 chip is just 512 1
// We load in the same way as with
// loading 16-bit words instead

```

Internal pins

Name	Value
11	0
o1[16]	0
12	0
o2[16]	0
13	0
o3[16]	21845
14	0
o4[16]	0
15	0
o5[16]	0
16	0
o6[16]	0
17	0
o7[16]	0

```

set load 1,
set address $B111101010,
set in $B0101010101010101,
tick,
output,
tock,
output;

set load 0,
set address $B000101010,
tick,
output;
tock,
output;
set address $B001101010,
eval,
output;
set address $B010101010,
eval,
output;
set address $B011101010,
eval,
output;
set address $B100101010,
eval,
output;
set address $B101101010,
eval,
output;
set address $B110101010,
eval,
output;
set address $B111101010,
eval,
output;

```

End of script - Comparison ended successfully



# CS 2200 HW 5 (Nand2Tetris 03)

RAM4K.hdl test:

Hardware Simulator (2.5) - D:\Projects\School\CS 2200\HW5\hds\CS2200-Nand2Tetris03\b\RAM4K.hdl

File View Run Help

Chip Name : RAM4K (Clocked) Time : 81

Input pins

Name	Value
in[16]	21845
load	0
address[12]	3925

Output pins

Name	Value
out[16]	21845

HDL

```

/**
 * Author:Sung Ki Ling
 * Date:2018/11/08
 * Memory of 4K registers, each
 * stored at the memory location
 * the in value is loaded into t
 * (the loaded value will be emi
 */
// The RAM4K chip is just 4K 16-
// We load in the same way as wi
// loading 16-bit words instead
// an address that tells which c

```

Internal pins

Name	Value
11	0
o1[16]	0
12	0
o2[16]	0
13	0
o3[16]	0
14	0
o4[16]	0
15	0
o5[16]	0
16	0
o6[16]	21845
17	0
o7[16]	0

```

set load 1,
set address $B111101010101,
set in $B01010101010101,
tick,
output,
tock,
output;

set load 0,
set address $B000101010101,
tick,
output;
tock,
output;
set address $B001101010101,
eval,
output;
set address $B010101010101,
eval,
output;
set address $B011101010101,
eval,
output;
set address $B100101010101,
eval,
output;
set address $B101101010101,
eval,
output;
set address $B110101010101,
eval,
output;
set address $B111101010101,
eval,
output;

```

End of script - Comparison ended successfully



RAM16K.hdl test:

Hardware Simulator (2.5) - D:\Projects\School\CS 2200\HW5\hds\CS2200-Nand2Tetris03\b\RAM16K.hdl

File View Run Help

Chip Name: RAM16K (Clocked) Time: 81

Input pins		Output pins	
Name	Value	Name	Value
in[16]	21845	out[16]	21845
load	0		
address[14]	15701		

Internal pins	
Name	Value
11	0
12	0
13	0
14	0
o1[16]	0
o2[16]	21845
o3[16]	0
o4[16]	0

HDL

```

/**
 * Author:Christopher O'Toole
 * Date:2018/11/08
 * Memory of 16K registers, each
 * stored at the memory location
 * the in value is loaded into t
 * (the loaded value will be emi
 */

// The RAM16K chip can set or re
// load=0 then the value current
// in in[16] is set.

```

```

set load 1,
set address $B11110101010101,
set in $B0101010101010101,
tick,
output,
tock,
output;

set load 0,
set address $B00010101010101,
tick,
output;
tock,
output;
set address $B00110101010101,
eval,
output;
set address $B01010101010101,
eval,
output;
set address $B01110101010101,
eval,
output;
set address $B10010101010101,
eval,
output;
set address $B10110101010101,
eval,
output;
set address $B11010101010101,
eval,
output;
set address $B11110101010101,
eval,
output;

```

End of script - Comparison ended successfully

PC.hdl test:

Hardware Simulator (2.5) - D:\Projects\School\CS 2200\HW5\hdl\CS2200-Nand2Tetris03\PC.hdl

File View Run Help

Chip Name: **PC (Clocked)** Time: **15**

Input pins		Output pins	
Name	Value	Name	Value
in[16]	22222	out[16]	0
load	0		
inc	0		
reset	1		

Internal pins	
Name	Value
cur[16]	0
incCur[16]	1
val2[16]	0
val3[16]	0
val1[16]	0

**HDL**

```

/**
 * Author:Yunchao Zhang
 * Date:2018-11-09
 * A 16-bit counter with load and increment
 * if (reset[t] == 1) out[t] = 0;
 * else if (load[t] == 1) out[t] = loadVal;
 * else if (inc[t] == 1) out[t] = (out[t] + 1) % 256;
 * else out[t] = out[t];
 */

// The program counter is a register
// executed at the current time.
// After each instruction is fetched

```

**Script**

```

tock,
output;

set reset 1,
tick,
output;

tock,
output;

set in 0,
set reset 0,
set load 1,
tick,
output;

tock,
output;

set load 0,
set inc 1,
tick,
output;

set in 22222,
set reset 1,
set inc 0,
tick,
output;

tock,
output;

```

End of script - Comparison ended successfully