CS 2200 HW 5 (Nand2Tetris 03)

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 at 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.

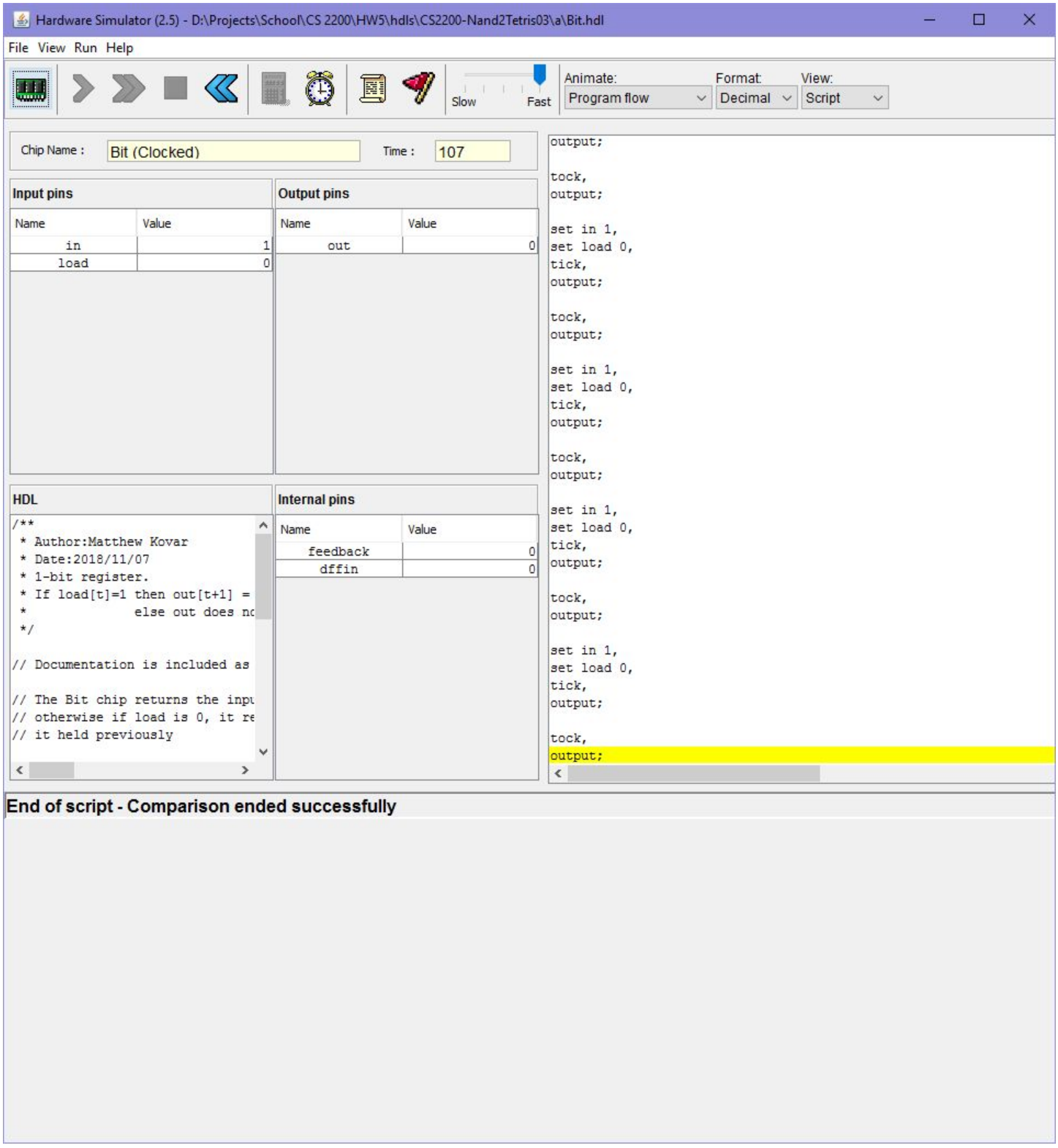
The RAM4K chip is just eight RAM512 units tied together, but now with a 12-bit address provided, since 2^12 = 4096. This chip functions identically to the RAM512 unit, except now up to 4096 individual 16-bit words can be stored instead of only 512.

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.

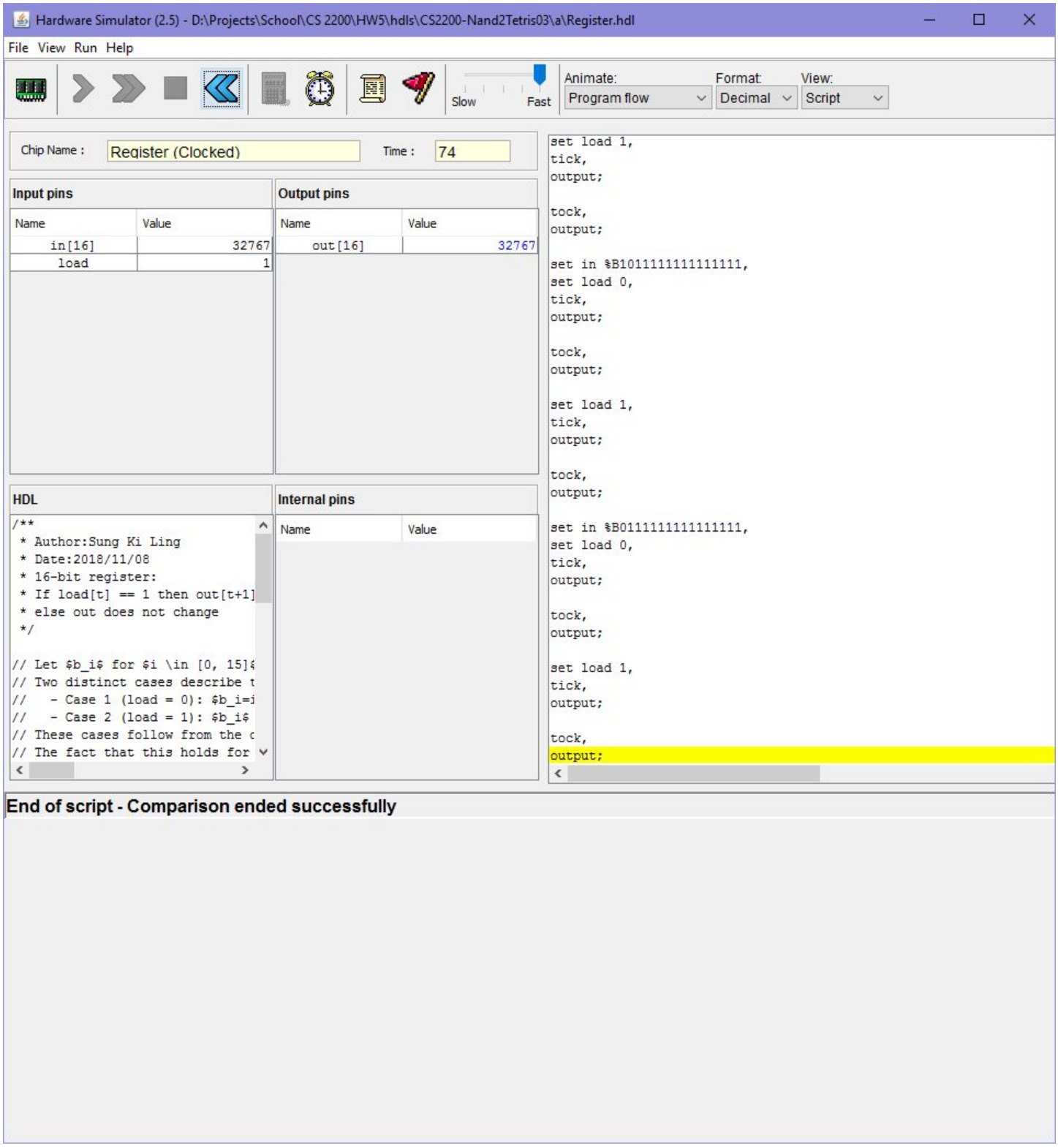
The program counter is a register that contains the address (location) of the instruction being executed at the current time. As each instruction, the program counter increases its stored value by 1. After each instruction is fetched, the program counter points to the next instruction in the sequence. 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:

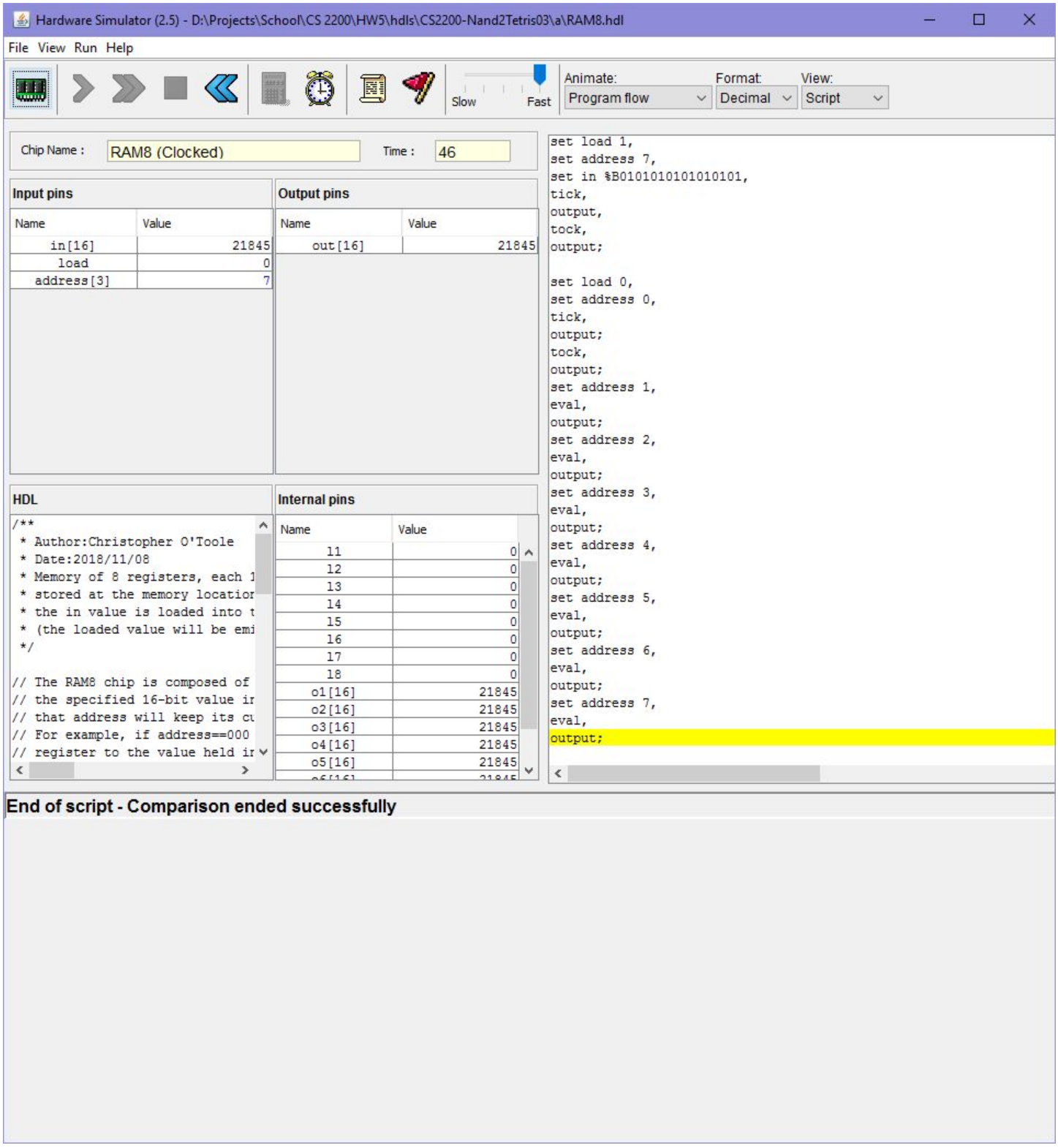
Bit.hdl test:


Hardware Simulator (2.5) - D:\Projects\School\CS 2200\HW5\hdls\CS2200-Nand2Tetris03\a\Bit.hdl
File View Run Help
Slow
Fast
Animate:
Program flow
Format:
Decimal
View:
Script
Chip Name :
Bit (Clocked)
Time :
107
Input pins
Name
Value
in
1
load
0
Output pins
Name
Value
out
0
HDL
/**
* Author:Matthew Kovar
* Date:2018/11/07
* 1-bit register.
* If load[t]=1 then out[t+1] =
* else out does nc
*/
// Documentation is included as
// The Bit chip returns the inpu
// otherwise if load is 0, it re
// it held previously
Internal pins
Name
Value
feedback
0
dffin
0
output;
tock,
output;
set in 1,
set load 0,
tick,
output;
tock,
output;
set in 1,
set load 0,
tick,
output;
tock,
output;
set in 1,
set load 0,
tick,
output;
tock,
output;
set in 1,
set load 0,
tick,
output;
tock,
output;
End of script - Comparison ended successfully

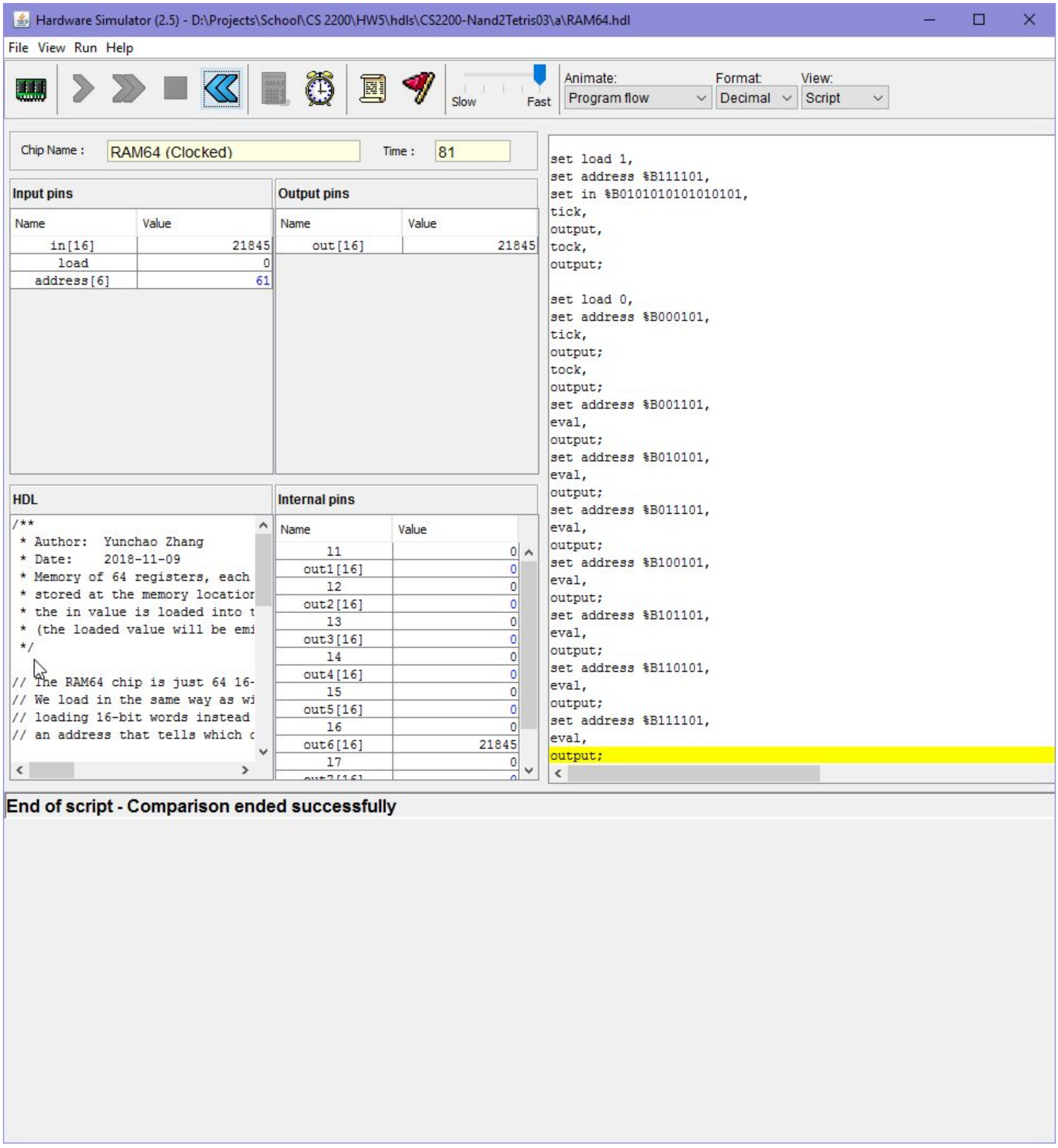
Register.hdl test:



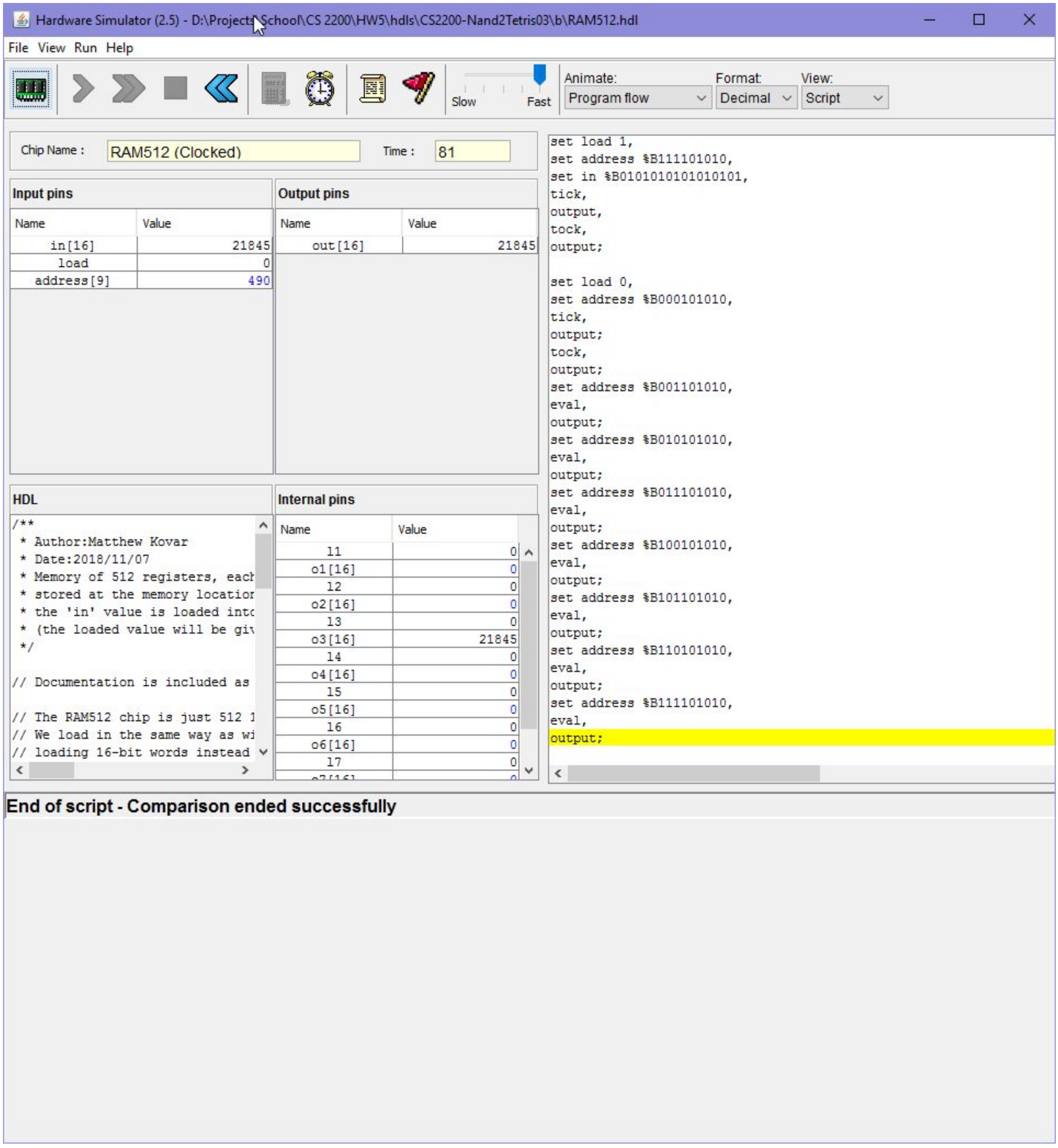
RAM8.hdl test:



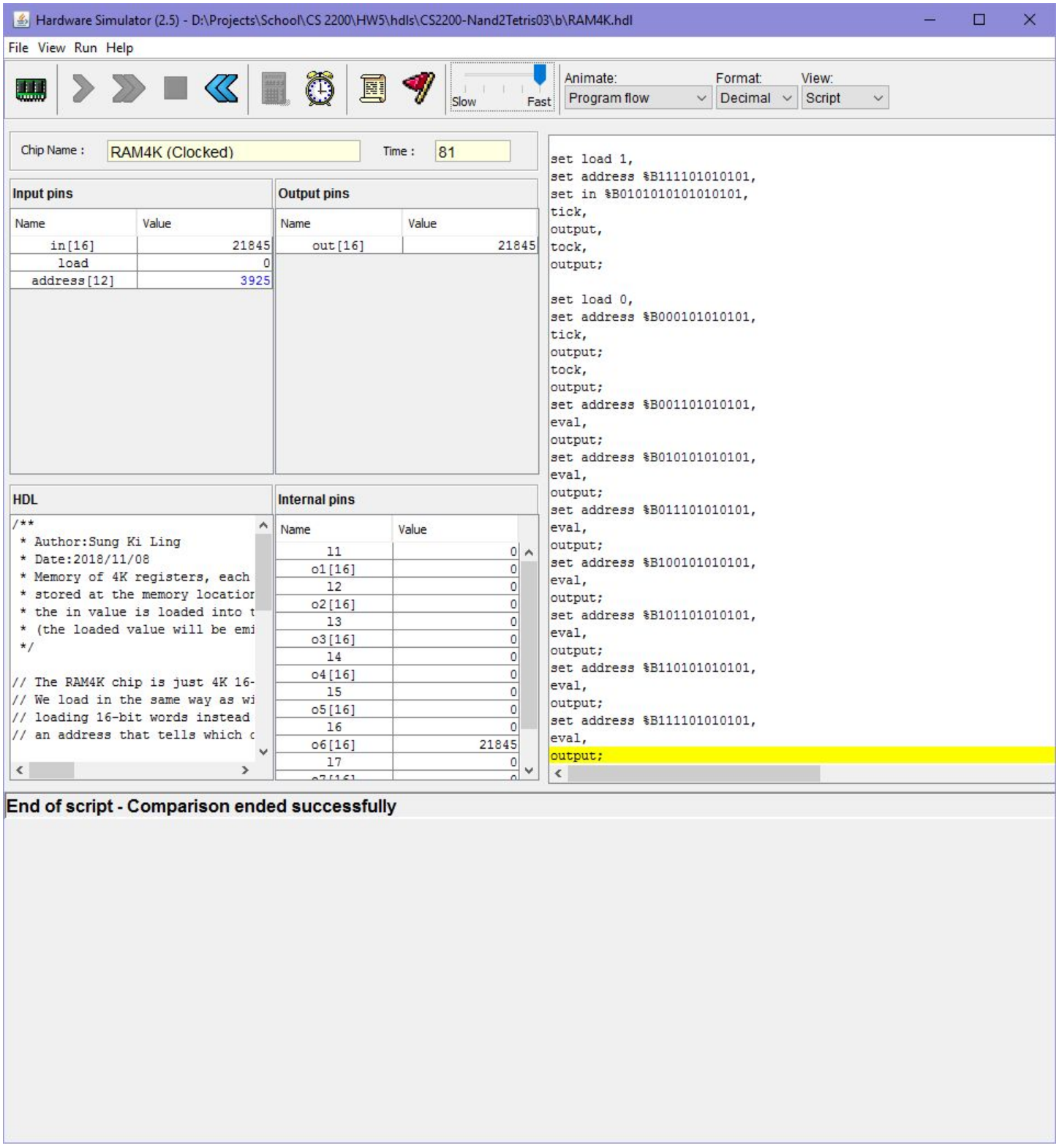
RAM64.hdl test:



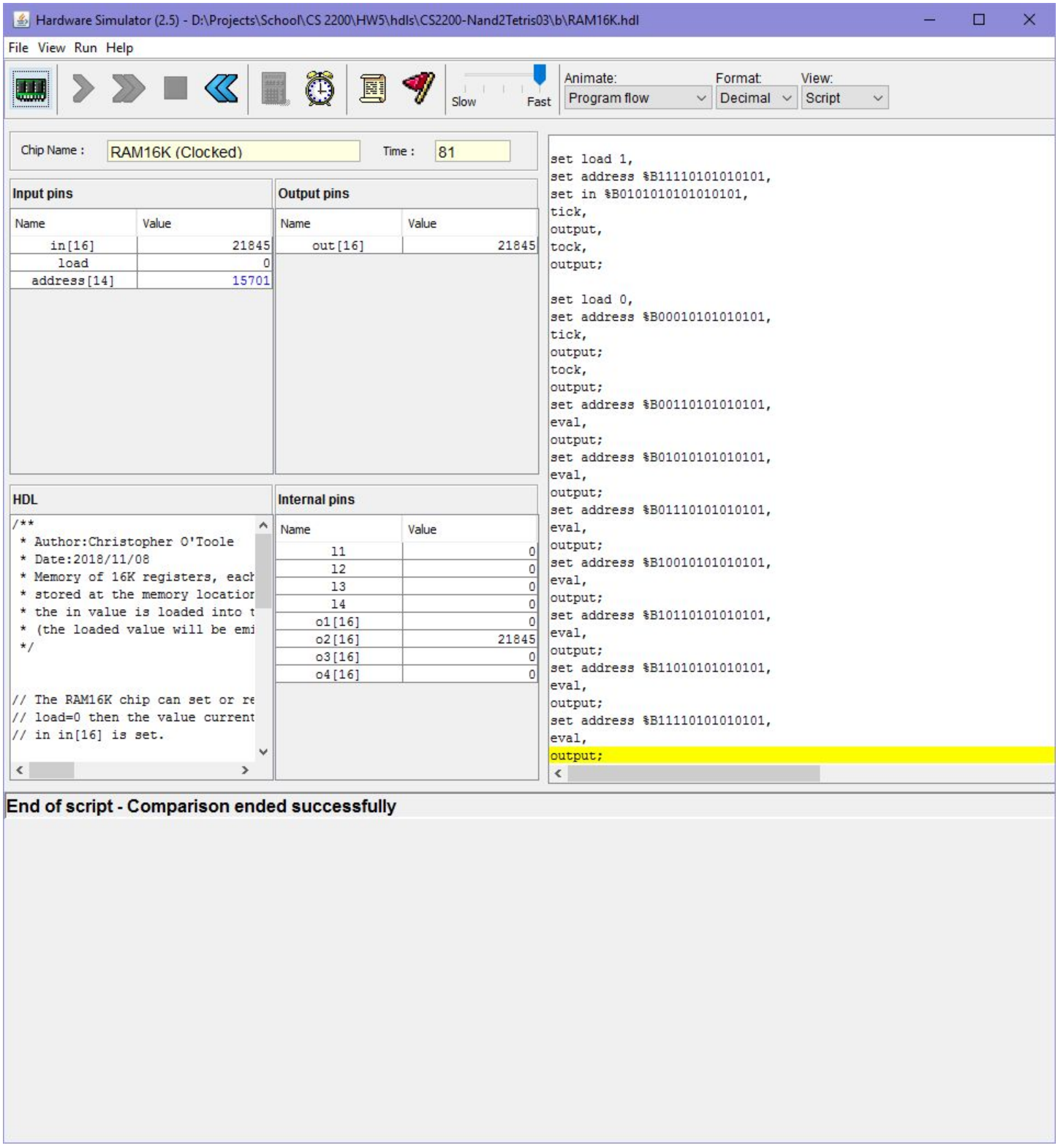
RAM512.hdl test:



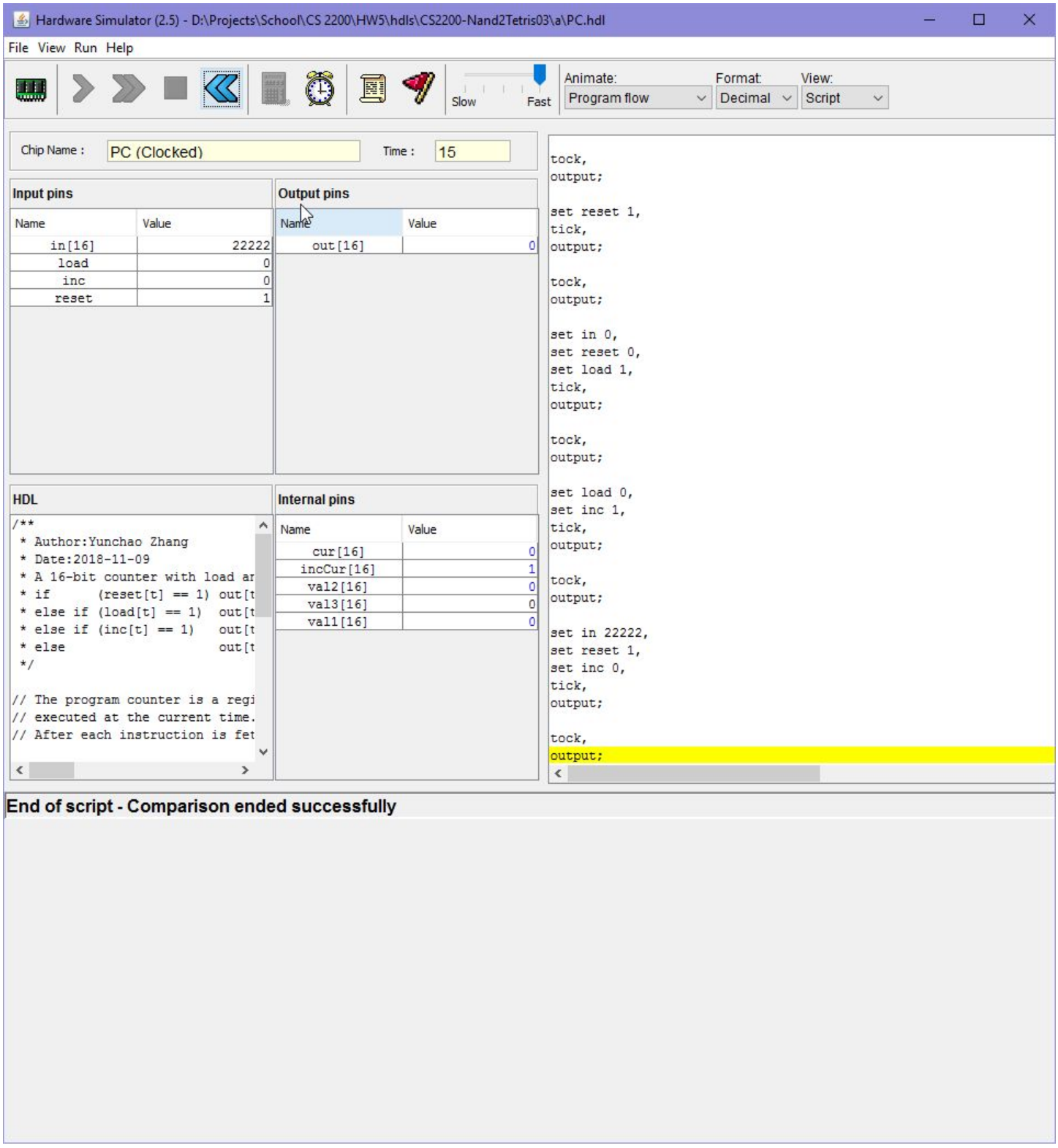
RAM4K.hdl test:



RAM16K.hdl test:



PC.hdl test:


Hardware Simulator (2.5) - D:\Projects\School\CS 2200\HW5\hdls\CS2200-Nand2Tetris03\a\PC.hdl
File View Run Help
Slow
Fast
Animate:
Program flow
Format:
Decimal
View:
Script
Chip Name :
PC (Clocked)
Time :
15
Input pins
Name
Value
in[16]
22222
load
0
inc
0
reset
1
Output pins
Name
Value
out[16]
0
HDL
/**
* Author:Yunchao Zhang
* Date:2018-11-09
* A 16-bit counter with load ar
* if (reset[t] == 1) out[t
* else if (load[t] == 1) out[t
* else if (inc[t] == 1) out[t
* else out[t
*/
// The program counter is a regi
// executed at the current time.
// After each instruction is fet
Internal pins
Name
Value
cur[16]
0
incCur[16]
1
val2[16]
0
val3[16]
0
val1[16]
0
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;
tock,
output;
set in 22222,
set reset 1,
set inc 0,
tick,
output;
tock,
output;
End of script - Comparison ended successfully