DMux4Way(in=load, sel=address[13..14], a=loadram1, b=loadram2, c=loadscreen, d=loadkbd);

DMux4Way: This line declares a 4-way Demultiplexer, which takes a control input load and a 2-bit selection input address[13..14]. It produces four outputs (loadram1, loadram2, loadscreen, and loadkbd). The demultiplexer directs the load signal to one of its outputs based on the value of the selection bits.

Or(a=loadram1, b=loadram2, out=loadram);

Or: This line declares an OR gate that takes two inputs (loadram1 and loadram2) and produces an output (loadram). The purpose of this gate is to combine the outputs of the 4-way demultiplexer to create a unified load signal for the entire memory system.

RAM16K(in=in, load=loadram, address=address[0..13], out=ramout);

RAM16K: This line declares a 16K RAM (Random Access Memory) chip. It has inputs for data (in), load signal (loadram), and a 14-bit address input (address[0..13]). The output of the RAM chip is ramout. This component represents the main memory storage where data is read from or written to based on the load signal and address.

Screen(in=in, load=loadscreen, address=address[0..12], out=scrout);

Screen: This line declares a component representing a computer screen. It takes inputs for data (in), load signal (loadscreen), and a 13-bit address input (address[0..12]). The output is scrout, which represents the content of the screen memory. This allows the system to display data on the screen.

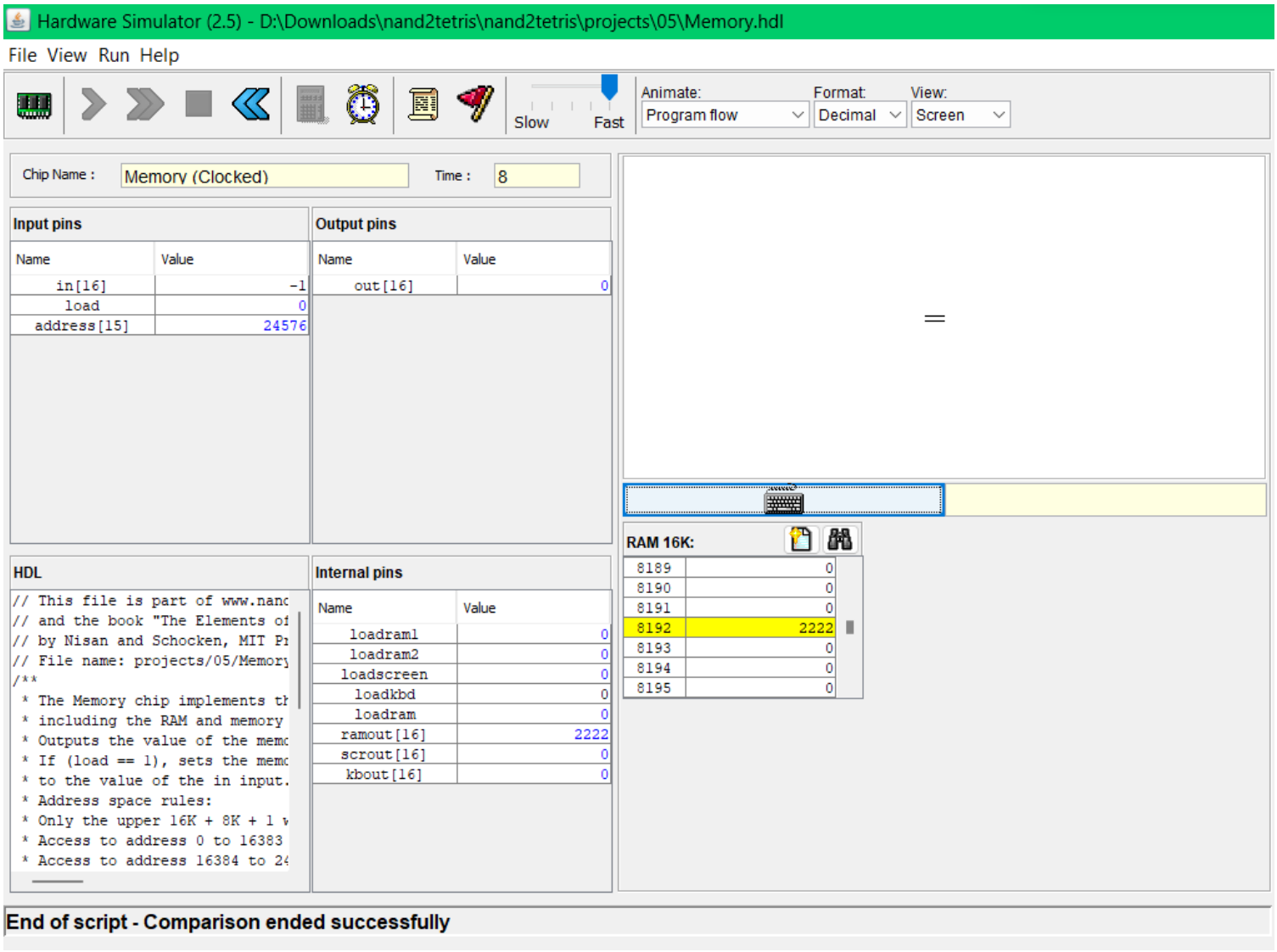
Keyboard(out=kbout);

Keyboard: This line declares a component representing a computer keyboard. It has no inputs, and its output is kbout, which represents the state of the keyboard. This allows the system to read input from the keyboard.

Mux4Way16(a=ramout, b=ramout, c=scrout, d=kbout, sel=address[13..14], out=out);

Mux4Way16: This line declares a 4-way 16-bit multiplexer. It takes four 16-bit inputs (ramout, ramout, scrout, and kbout), a 2-bit selection input (address[13..14]), and produces a 16-bit output (out). The multiplexer selects one of the inputs based on the value of the selection bits. This component serves to choose the appropriate output for the memory system based on the address and is crucial for connecting the different memory components.

In summary, the Memory.hdl file combines various components (demultiplexer, OR gate, RAM, Screen, Keyboard, and multiplexer) to create a memory system that can handle different types of memory (RAM, Screen, and Keyboard) and route data appropriately based on the address and control signals. The demultiplexer helps direct the load signal to the correct memory component, and the multiplexer ensures the correct output is selected based on the address.


Hardware Simulator (2.5) - D:\Downloads\nand2tetris\nand2tetris\projects\05\Memory.hdl
File View Run Help
Slow
Fast
Animate:
Program flow
Format:
Decimal
View:
Screen
Chip Name :
Memory (Clocked)
Time :
8
Input pins
Name
Value
in[16]
-1
load
0
address[15]
24576
Output pins
Name
Value
out[16]
0
RAM 16K:
8189
0
8190
0
8191
0
8192
2222
8193
0
8194
0
8195
0
HDL
// This file is part of www.nanc
// and the book "The Elements of
// by Nisan and Schocken, MIT Pr
// File name: projects/05/Memory
/**
* The Memory chip implements th
* including the RAM and memory
* Outputs the value of the memo
* If (load == 1), sets the memo
* to the value of the in input.
* Address space rules:
* Only the upper 16K + 8K + 1 w
* Access to address 0 to 16383
* Access to address 16384 to 24
Internal pins
Name
Value
loadram1
0
loadram2
0
loadscreen
0
loadkbd
0
loadram
0
ramout[16]
2222
scrout[16]
0
kbout[16]
0
End of script - Comparison ended successfully