

MEMORY CHIP CODE:

```
CHIP Memory {
    IN in[16], load, address[15];
    OUT out[16];

    PARTS:

        //RAM16: 0 - 16383 in decimal
        //RAM16: 0000000000000000 - 0011111111111111 in 2's complement

        //SCREEN: 16384 - 24575 in decimal
        //SCREEN: 0100000000000000 - 0101111111111111 in 2's complement

        //KEYBOARD: 24576 - onwards
        //KEYBOARD: 0110000000000000 - 0111111111111111 in 2's complement

        //To see if I need to access memory in RAM16, I don't need to look at
the last 2 bits in address: address[0..13]
        //To see if I need to access memory in SCREEN, I don't need to look at
the last 3 bits in address: address[0..12]

        //From this I can use the excluded bits as a way to split up my
options and organize the 3 categories of memory:
        //00-> RAM16K
        //01-> RAM16K
        //10-> SCREEN
        //11-> KEYBOARD

        //These values are all from address[13..14]. I don't care about
address[15] because this tells me if its negative or not.
        //To combine my two buses of RAM16K, an OR gate can be used

        DMux4Way(in=load, sel=address[13..14], a=RAMld1, b=RAMld2, c=SCRld,
d=KBDld);

        Or(a=RAMld1, b=RAMld2, out=RAMld);

        RAM16K(in=in, load=RAMld, address=address[0..13], out=RAMout);
        Screen(in=in, load=SCRld, address=address[0..12], out=SCRout);
        Keyboard(out=KBDout);

        Mux4Way16(a=RAMout, b=RAMout, c=SCRout, d=KBDout, sel=address[13..14],
out=out);
}
```

MEMORY CHIP TEST:

Hardware Simulator (2.5) - C:\Users\kyleg\OneDrive\Desktop\CS271\nand2tetris_files\nand2tetris_files\02_software\nand2tetris\projects\05\Memory.hdl

File View Run Help

The screenshot shows the Hardware Simulator (2.5) interface. The top toolbar includes icons for running, pausing, and stepping through the simulation, along with a speed slider (Slow to Fast) and dropdown menus for animation (Program flow), format (Decimal), and view (Screen). The main window displays the 'Memory (Clocked)' chip. The 'Input pins' table shows 'in[16]' at -1, 'load' at 0, and 'address[15]' at 24576. The 'Output pins' table shows 'out[16]' at 0. The 'HDL' section contains the Verilog code for the memory chip. The 'Internal pins' table shows various internal signals, with 'RAMout[16]' at 2222. The 'RAM 16K' table shows the contents of memory locations 8189 to 8195, with location 8192 containing the value 2222. The status bar at the bottom indicates 'End of script - Comparison ended successfully'.

Input pins		Output pins	
Name	Value	Name	Value
in[16]	-1	out[16]	0
load	0		
address[15]	24576		

Internal pins	
Name	Value
RAM1d1	0
RAM1d2	0
SCR1d	0
KBD1d	0
RAM1d	0
RAMout[16]	2222
SCRout[16]	0
KBDout[16]	0

RAM 16K:	
8189	0
8190	0
8191	0
8192	2222
8193	0
8194	0
8195	0

End of script - Comparison ended successfully

CPU CHIP:

CHIP CPU {

```
IN inM[16],           // M value input (M = contents of RAM[A])
  instruction[16],    // Instruction for execution
  reset;              // Signals whether to re-start the current
                      // program (reset==1) or continue executing
                      // the current program (reset==0).
```

```
OUT outM[16],         // M value output
  writeM,              // Write to M?
  addressM[15],        // Address in data memory (of M)
  pc[15];              // address of next instruction
```

PARTS:

```

// Put your code here:
Not(in=instruction[15], out=Ainstr); //want negation because
instruction[15] being 0 means loads A so 0 needs to become 1 for Muxer
Mux16(a=ALUout, b=instruction, sel=Ainstr, out=Ain);

Or(a=Ainstr, b=instruction[5], out=Aload); //to write to A, either
op-code is 0 (A-instruction), or instruction[5]
And(a=instruction[15], b=instruction[4], out=Dload); //to write to D,
op-code will be 1, and instruction[4]
And(a=instruction[15], b=instruction[3], out=writeM); //to write to M,
opt-code will be 1, and instruction[3]

ARegister(in=Ain, load=Aload, out=Aout, out[0..14]=addressM);
Mux16(a=Aout, b=inM, sel=instruction[12], out=ALUin2);
DRegister(in=ALUout, load=Dload, out=ALUin1);

ALU(x=ALUin1, y=ALUin2, zx=instruction[11], nx=instruction[10],
zy=instruction[9], ny=instruction[8], f=instruction[7], no=instruction[6],
out=outM, out=ALUout, zr=outzr, ng=outng);

//To use logic for PC counter, I need negations of zr, ng, j1, j2, and
j3
Not(in=outzr, out=notoutzr);
Not(in=outng, out=notoutng);
Not(in=instruction[2], out=notj1);
Not(in=instruction[1], out=notj2);
Not(in=instruction[0], out=notj3);

//JGT: Not(j1) And Not(j2) And j3 And Not(zr) and Not(ng)
And(a=instruction[15], b=notj1, out=a0); //First need to check to
make sure it is a C-instruction
And(a=a0, b=notj2, out=a1);
And(a=a1, b=instruction[0], out=a2);
And(a=a2, b=notoutzr, out=a3);
And(a=a3, b=notoutng, out=jgt);

//JEQ: Not(j1) And j2 And Not(j3) And zr And Not(ng)
And(a=instruction[15], b=notj1, out=b0); //First need to check to
make sure it is a C-instruction
And(a=b0, b=instruction[1], out=b1);

```

```

And(a=b1, b=notj3, out= b2);
And(a=b2, b=outzr, out=b3);
And(a=b3, b=notoutng, out=ieq);

//JGE: Not(j1) And j2 And j3 And (zr Or Not(zr)) And Not(ng)
And(a=instruction[15], b=notj1, out=c0); //First need to check to
make sure it is a C-instruction
And(a=c0, b=instruction[1], out=c1);
And(a=c1, b=instruction[0], out=c2);
And(a=c2, b=outzr, out=c3);
And(a=c2, b=notoutzr, out=c4);
Or(a=c3, b=c4, out=c5);
And(a=c5, b=notoutng, out=jge);

//JLT: j1 And Not(j2) And Not(j3) And Not(zr) And ng
And(a=instruction[15], b=instruction[2], out=d0); //First need to
check to make sure it is a C-instruction
And(a=d0, b=notj2, out=d1);
And(a=d1, b=notj3, out=d2);
And(a=d2, b=notoutzr, out=d3);
And(a=d3, b=outng, out=jlt);

//JNE: j1 And Not(j2) and j3 and Not(zr) and (ng Or Not(ng))
And(a=instruction[15], b=instruction[2], out=e0); //First need to
check to make sure it is a C-instruction
And(a=e0, b=notj2, out=e1);
And(a=e1, b=instruction[0], out=e2);
And(a=e2, b=notoutzr, out=e3);
And(a=e3, b=outng, out=e4);
And(a=e3, b=notoutng, out=e5);
Or(a=e4, b=e5, out=jne);

//JLE: j1 And j2 And Not(j3) And (((Not(zr) And ng)) Or (zr And
Not(ng)))
And(a=instruction[15], b=instruction[2], out=f0); //First need to
check to make sure it is a C-instruction
And(a=f0, b=instruction[1], out=f1);
And(a=f1, b=notj3, out=f2);
And(a=f2, b=outzr, out=f3);
And(a=f3, b=notoutng, out=f4);

```

```

And(a=f2, b=notoutzr, out=f5);
And(a=f5, b=outnq, out=f6);
Or(a=f4, b=f6, out=jle);

//JMP j1 And j2 And j3
And(a=instruction[15], b=instruction[2], out=term0); //First need to
check to make sure it is a C-instruction
And(a=term0, b=instruction[1], out=term1);
And(a=term1, b=instruction[0], out=jump);

//if any options are 1, then jump
Or(a=jgt, b=jeq, out=term2);
Or(a=term2, b=jge, out=term3);
Or(a=term3, b=jlt, out=term4);
Or(a=term4, b=jne, out=term5);
Or(a=term5, b=jle, out=term6);
Or(a=term6, b=jump, out=pcload);

PC(in=Aout, load=pcload, inc=true, reset=reset, out[0..14]=pc);
}

```

CPU CHIP TEST:

File View Run Help

Chip Name: CPU (Clocked) Time: 48

Input pins		Output pins	
Name	Value	Name	Value
inM[16]	11111	outM[16]	1
instruction[16]	32767	writeM	0
reset	0	addressM[15]	32767
		pc[15]	1

Internal pins	
Name	Value
Ainstr	1
ALUout[16]	1
Ain[16]	32767
Aload	1
Dload	0
Aout[16]	32767
ALUin2[16]	11111
ALUin1[16]	1
outzr	0
outng	0
notoutzr	1
notoutng	1
notjl	0

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/05/CPU.hdl

/**
 * The Hack CPU (Central Processor)
 * two registers named A and D,
 * The CPU is designed to fetch
 * the Hack machine language. It
 * Executes the inputted instruction
 * language specification. The
 * refer to CPU-resident registers
 * memory location addressed by
 */

set instruction $B1110001100000110, // D:JNE
tick, output, tock, output;

set instruction $B1110001100000111, // D:JMP
tick, output, tock, output;

set instruction $B1110111111010000, // D=1
tick, output, tock, output;

set instruction $B1110001100000001, // D:JGT
tick, output, tock, output;

set instruction $B1110001100000010, // D:JEQ
tick, output, tock, output;

set instruction $B1110001100000011, // D:JGE
tick, output, tock, output;

set instruction $B1110001100000100, // D:JLT
tick, output, tock, output;

set instruction $B1110001100000101, // D:JNE
tick, output, tock, output;

set instruction $B1110001100000110, // D:JLE
tick, output, tock, output;

set instruction $B1110001100000111, // D:JMP
tick, output, tock, output;

set reset 1;
tick, output, tock, output;

set reset 0;
tick, output, tock, output;
```

End of script - Comparison ended successfully

CPU CHIP EXTERNAL TEST:

Hardware Simulator (2.5) - C:\Users\kyleg\OneDrive\Desktop\CS271\nand2tetris_files\nand2tetris_files\02_software\nand2tetris\projects\05\CPU.hdl

File View Run Help

Chip Name: CPU (Clocked) Time: 46

Input pins		Output pins	
Name	Value	Name	Value
inM[16]	11111	outM[16]	1
instruction[16]	32767	writeM	0
reset	0	addressM[15]	32767
		pc[15]	1

Internal pins	
Name	Value
Ainstr	1
ALUout[16]	1
Ain[16]	32767
Aload	1
Dload	0
Aout[16]	32767
ALUin2[16]	11111
ALUin1[16]	1
outzr	0
outng	0
notoutzr	1
notoutng	1
notj1	0

```

HDL
And(a=instruction[15], b=not
And(a=a0, b=notj2, out=a1);
And(a=a1, b=instruction[0],
And(a=a2, b=notoutzr, out=a3;
And(a=a3, b=notoutng, out=j2;

//JEQ: Not(j1) And j2 And No
And(a=instruction[15], b=not
And(a=b0, b=instruction[1],
And(a=b1, b=notj3, out= b2);
And(a=b2, b=outzr, out=b3);
And(a=b3, b=notoutng, out=j3;

//JGE: Not(j1) And j2 And j3;

set instruction $B1110001100000110, // D:JLE
tick, output, tock, output;

set instruction $B1110001100000111, // D:JMP
tick, output, tock, output;

set instruction $B1110111111010000, // D=1
tick, output, tock, output;

set instruction $B1110001100000001, // D:JGT
tick, output, tock, output;

set instruction $B1110001100000010, // D:JEQ
tick, output, tock, output;

set instruction $B1110001100000011, // D:JGE
tick, output, tock, output;

set instruction $B1110001100000100, // D:JLT
tick, output, tock, output;

set instruction $B1110001100000101, // D:JNE
tick, output, tock, output;

set instruction $B1110001100000110, // D:JLE
tick, output, tock, output;

set instruction $B1110001100000111, // D:JMP
tick, output, tock, output;

set reset 1;
tick, output, tock, output;

set reset 0;
tick, output, tock, output;

```

End of script - Comparison ended successfully

COMPUTER CHIP:

CHIP Computer {

```

    IN reset;

    PARTS:
        // Put your code here:
        ROM32K(address=pc, out=instruction);
        CPU(inM=out1, instruction=instruction, reset=reset, outM=outM,
writeM=loadM, addressM=addressM, pc=pc);
        Memory(in=outM, load=loadM, address=addressM, out=out1, out=out);
}

```

COMPUTER CHIP ADD TEST:

Hardware Simulator (2.5) - C:\Users\kyleg\OneDrive\Desktop\CS271\nand2tetris_files\nand2tetris_files\02_software\nand2tetris\projects\05\Computer.

File View Run Help

Chip Name: **Computer (Clocked)** Time: **13**

Input pins		Output pins	
Name	Value	Name	Value
reset	0		

Internal pins	
Name	Value
pc[15]	6
instruction[16]	0
outI[16]	5
outM[16]	0
loadM	0
addressM[15]	0
out[16]	5

HDL

```
* screen may show some output :
* with the computer via the key
*/
CHIP Computer {
    IN reset;

    PARTS:
        // Put your code here:
        ROM32K(address=pc, out=inst:
        CPU(inM=outI, instruction=ir
        Memory(in=outM, load=loadM,
}

// 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/05/ComputerAdd.tst

load Computer.hdl,
output-file ComputerAdd.out,
compare-to ComputerAdd.cmp,
output-list time$S1.4.1 reset$B2.1.2 ARegister[0]$D1.7.1 DRegister[0]$D1.

// Load a program written in the Hack machine language.
// The program adds the two constants 2 and 3 and writes the result in RAM
ROM32K load Add.hack,
output;

// First run (at the beginning PC=0)
repeat 6 {
    tick, tock, output;
}

// Reset the PC
set reset 1,
set RAM16K[0] 0,
tick, tock, output;

// Second run, to check that the PC was reset correctly.
set reset 0,
repeat 6 {
    tick, tock, output;
}
```

End of script - Comparison ended successfully

COMPUTER CHIP ADD EXTERNAL TEST:

File View Run Help

Chip Name: **Computer (Clocked)** Time: **25**

Animate: **No animation** Format: **Decimal** View: **Script**

Slow Fast

Input pins		Output pins	
Name	Value	Name	Value
reset	0		

HDL

```

* screen may show some output :
* with the computer via the key
*/
CHIP Computer {
    IN reset;

    PARTS:
        // Put your code here:
        ROM32K(address=pc, out=inst;
        CPU(inM=outI, instruction=ir
        Memory(in=outM, load=loadM,
    }
  
```

Internal pins	
Name	Value
pc[15]	14
instruction[16]	14
outI[16]	23456
outM[16]	0
loadM	0
addressM[15]	2
out[16]	23456

```

// file name: projects/05/ComputerMax.tst
load Computer.hdl,
output-file ComputerMax.out,
compare-to ComputerMax.cmp,
output-list time%S1.4.1 reset%B2.1.2 ARegister[]%D1.7.1 DRegister[]%D1

// Load a program written in the Hack machine language.
// The program computes the maximum of RAM[0] and RAM[1]
// and writes the result in RAM[2].

ROM32K load Max.hack,

// first run: compute max(3,5)
set RAM16K[0] 3,
set RAM16K[1] 5,
output;

repeat 14 {
    tick, tock, output;
}

// reset the PC
set reset 1,
tick, tock, output;

// second run: compute max(23456,12345)
set reset 0,
set RAM16K[0] 23456,
set RAM16K[1] 12345,
output;

// The run on these inputs needs less cycles (different branching)
repeat 10 {
    tick, tock, output;
}
  
```

End of script - Comparison ended successfully

COMPUTER CHIP MAX EXTERNAL TEST:

File View Run Help

Chip Name: **Computer (Clocked)** Time: **25**

Animate: **No animation** Format: **Decimal** View: **Script**

Slow Fast

Input pins		Output pins	
Name	Value	Name	Value
reset	0		

HDL

```

* screen may show some output &
* with the computer via the key
*/
CHIP Computer {
    IN reset;

    PARTS:
        // Put your code here:
        ROM32K(address=pc, out=instr;
        CPU(inM=outI, instruction=ir;
        Memory(in=outM, load=loadM,
}

```

Internal pins

Name	Value
pc[15]	14
instruction[16]	14
outI[16]	23456
outM[16]	0
loadM	0
addressM[15]	2
out[16]	23456

```

// by Nisan and Schocken, MIT Press.
// File name: projects/05/ComputerMax-external.tst

load Computer.hdl,
output-file ComputerMax-external.out,
compare-to ComputerMax-external.cmp,
output-list time$S1.4.1 reset$B2.1.2 RAM16K[0]$D1.7.1 RAM16K[1]$D1.7.1

// Load a program written in the Hack machine language.
// The program computes the maximum of RAM[0] and RAM[1]
// and writes the result in RAM[2].
ROM32K load Max.hack,

// first run: compute max(3,5)
set RAM16K[0] 3,
set RAM16K[1] 5,
output;

repeat 14 {
    tick, tock, output;
}

// reset the PC
set reset 1,
tick, tock, output;

// second run: compute max(23456,12345)
set reset 0,
set RAM16K[0] 23456,
set RAM16K[1] 12345,
output;

// The run on these inputs needs less cycles (different branching)
repeat 10 {
    tick, tock, output;
}

```

End of script - Comparison ended successfully

COMPUTER CHIP RECTANGLE TEST:

Hardware Simulator (2.5) - C:\Users\kyleg\OneDrive\Desktop\CS271\nand2tetris_files\nand2tetris_files\02_software\nand2tetris\projects\05\Computer.

File View Run Help

Chip Name: **Computer (Clocked)** Time: **63**

Input pins

Name	Value
reset	0

Output pins

Name	Value
------	-------

HDL

```

* screen may show some output :
* with the computer via the key
*/
CHIP Computer {
    IN reset;

    PARTS:
    // Put your code here:
    ROM32K(address=pc, out=inst);
    CPU(inM=outI, instruction=inst);
    Memory(in=outM, load=loadM,

```

Internal pins

Name	Value
pc[15]	24
instruction[16]	-5497
outI[16]	0
outM[16]	0
loadM	0
addressM[15]	23
out[16]	0

RAM 16K:

20	0
21	0
22	0
23	0
24	0
25	0
26	0

ROM:

Bin	Value
18	1110001100001000
19	0000000000010000
20	111110010011000
21	000000000001010
22	1110001100000001
23	0000000000010111
24	1110101010000111

ALU

D Input: 0

M/A Input: 23

ALU output: 0

End of script - Comparison ended successfully

COMPUTER CHIP EXTERNAL RECTANGLE TEST:

File View Run Help

Chip Name: **Computer (Clocked)** Time: **63**

Input pins

Name	Value
reset	0

Output pins

Name	Value
------	-------

HDL

```
* screen may show some output &
* with the computer via the key
*/
CHIP Computer {
    IN reset;

    PARTS:
    // Put your code here:
    ROM32K(address=pc, out=inst);
    CPU(inM=outI, instruction=ir,
    Memory(in=outM, load=loadM,
```

Internal pins

Name	Value
pc[15]	24
instruction[16]	-5497
outI[16]	0
outM[16]	0
loadM	0
addressM[15]	23
out[16]	0

RAM 16K:

20	0
21	0
22	0
23	0
24	0
25	0
26	0

ROM: Bin

18	1110001100001000
19	0000000000010000
20	1111110010011000
21	0000000000001010
22	1110001100000001
23	0000000000010111
24	1110101010000111

A: 23 **D:** 0 **PC:** 24

ALU

D Input: 0

M/A Input: 23

ALU output: 0

End of script - Comparison ended successfully