

```
CPU.hdl x Memory.hdl x Computer.hdl x
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/05/Computer.hdl
5
6 /**
7  * The Computer chip, i.e. the top-most chip of the Hack architecture.
8  * The Computer chip consists of CPU, ROM and RAM chip-parts.
9  * It is assumed that the ROM is pre-loaded with some Hack program.
10 * The Computer chip has a single 1-bit input, named "reset".
11 * When reset is 0, the stored program starts executing.
12 * When reset is 1, the program's execution restarts.
13 * Thus, to start a program's execution, reset must be pushed "up" (1)
14 * and "down" (0). From this point onward the user is at the mercy of
15 * the software. In particular, depending on the program loaded into
16 * the computer, the screen may show some output and the user may be
17 * expected to interact with the computer via the keyboard.
18 */
19
20 CHIP Computer {
21
22     IN reset;
23
24     PARTS:
25     ROM32K(address=pc, out=instruction);
26     CPU(inM=memOut, instruction=instruction, reset=reset, outM=outM,
27         writeM=writeM, addressM=addressM, pc=pc);
28     Memory(in=outM, load=writeM, address=addressM, out=memOut);
29 }
```



Fast

Script

Time : 13

Output pins

Name	Value

Internal pins

```


/**
 * The Computer chip, i.e. the t
 * The Computer chip consists of
 * It is assumed that the ROM is
 * The Computer chip has a singl
 * When reset is 0, the stored p
 * When reset is 1, the program'
 * Thus, to start a programâ€™s
 * and "down" (0). From this poi

```

Name	Value
pc[15]	6
instruction[16]	0
memOut[16]	5
outM[16]	0
writeM	0
addressM[15]	0

```
repeat 6 {
    tick, tock, output;
}
```

End of script - Comparison ended successfully



Animate: Program flow
 Format: Decimal
 View: Script

Slow Fast

Chip Name : Computer (Clocked) Time : 25

Input pins		Output pins	
Name	Value	Name	Value
reset	0		

HDL	Internal pins														
<pre>// 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/Computer.hdl /** * The Computer chip, i.e. the nand2tetris computer. * The Computer chip consists of a ROM, RAM, ALU, and * It is assumed that the ROM is initialized with the * The Computer chip has a single output pin, outM[16]. * When reset is 0, the stored program counter (PC) is * When reset is 1, the program counter (PC) is set to * Thus, to start a program, the user must set the PC * and "down" (0). From this point, the computer will</pre>	<table><tr><th>Name</th><th>Value</th></tr><tr><td>pc[15]</td><td>14</td></tr><tr><td>instruction[16]</td><td>14</td></tr><tr><td>memOut[16]</td><td>23456</td></tr><tr><td>outM[16]</td><td>0</td></tr><tr><td>writeM</td><td>0</td></tr><tr><td>addressM[15]</td><td>2</td></tr></table>	Name	Value	pc[15]	14	instruction[16]	14	memOut[16]	23456	outM[16]	0	writeM	0	addressM[15]	2
Name	Value														
pc[15]	14														
instruction[16]	14														
memOut[16]	23456														
outM[16]	0														
writeM	0														
addressM[15]	2														
<	>														

```
// File name: projects/05/ComputerMax.tst

load Computer.hdl,
output-file ComputerMax.out,
compare-to ComputerMax.cmp,
output-list time%$1.4.1 reset%B2.1.2 ARegister[]%D1.7.1 DRegister[]%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;

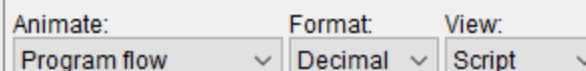
repeat 14 {
    tick, tock, output;
}

// 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












Time : 13


HDL	Internal pins														
<pre>// 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/Computer.hdl /** * The Computer chip, i.e. the Turing Machine. * The Computer chip consists of a ROM, a RAM, and a * It is assumed that the ROM is initialized with the * The Computer chip has a single output, 'outM'. * When reset is 0, the stored program counter is 0. * When reset is 1, the program counter is set to the * Thus, to start a program, the user must load the * and "down" (0). From this point, the program */</pre>	<table> <thead> <tr> <th>Name</th><th>Value</th></tr> </thead> <tbody> <tr> <td>pc[15]</td><td></td></tr> <tr> <td>instruction[16]</td><td></td></tr> <tr> <td>memOut[16]</td><td></td></tr> <tr> <td>outM[16]</td><td></td></tr> <tr> <td>writeM</td><td></td></tr> <tr> <td>addressM[15]</td><td></td></tr> </tbody> </table>	Name	Value	pc[15]		instruction[16]		memOut[16]		outM[16]		writeM		addressM[15]	
Name	Value														
pc[15]															
instruction[16]															
memOut[16]															
outM[16]															
writeM															
addressM[15]															

```
// 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-external.tst

load Computer.hdl,
output-file ComputerAdd-external.out,
compare-to ComputerAdd-external.cmp,
output-list time%S1.4.1 reset%B2.1.2 RAM16K[0]%D1.7.1 RAM16K[1]%D1.7.1 RAM16K[2]%D1.7.1 RAM16K[3]%D1.7.1 RAM16K[4]%D1.7.1 RAM16K[5]%D1.7.1 RAM16K[6]%D1.7.1 RAM16K[7]%D1.7.1 RAM16K[8]%D1.7.1 RAM16K[9]%D1.7.1 RAM16K[10]%D1.7.1 RAM16K[11]%D1.7.1 RAM16K[12]%D1.7.1 RAM16K[13]%D1.7.1 RAM16K[14]%D1.7.1 RAM16K[15]%D1.7.1 RAM16K[16]%D1.7.1 RAM16K[17]%D1.7.1 RAM16K[18]%D1.7.1 RAM16K[19]%D1.7.1 RAM16K[20]%D1.7.1 RAM16K[21]%D1.7.1 RAM16K[22]%D1.7.1 RAM16K[23]%D1.7.1 RAM16K[24]%D1.7.1 RAM16K[25]%D1.7.1 RAM16K[26]%D1.7.1 RAM16K[27]%D1.7.1 RAM16K[28]%D1.7.1 RAM16K[29]%D1.7.1 RAM16K[30]%D1.7.1 RAM16K[31]%D1.7.1 RAM16K[32]%D1.7.1 RAM16K[33]%D1.7.1 RAM16K[34]%D1.7.1 RAM16K[35]%D1.7.1 RAM16K[36]%D1.7.1 RAM16K[37]%D1.7.1 RAM16K[38]%D1.7.1 RAM16K[39]%D1.7.1 RAM16K[40]%D1.7.1 RAM16K[41]%D1.7.1 RAM16K[42]%D1.7.1 RAM16K[43]%D1.7.1 RAM16K[44]%D1.7.1 RAM16K[45]%D1.7.1 RAM16K[46]%D1.7.1 RAM16K[47]%D1.7.1 RAM16K[48]%D1.7.1 RAM16K[49]%D1.7.1 RAM16K[50]%D1.7.1 RAM16K[51]%D1.7.1 RAM16K[52]%D1.7.1 RAM16K[53]%D1.7.1 RAM16K[54]%D1.7.1 RAM16K[55]%D1.7.1 RAM16K[56]%D1.7.1 RAM16K[57]%D1.7.1 RAM16K[58]%D1.7.1 RAM16K[59]%D1.7.1 RAM16K[60]%D1.7.1 RAM16K[61]%D1.7.1 RAM16K[62]%D1.7.1 RAM16K[63]%D1.7.1 RAM16K[64]%D1.7.1 RAM16K[65]%D1.7.1 RAM16K[66]%D1.7.1 RAM16K[67]%D1.7.1 RAM16K[68]%D1.7.1 RAM16K[69]%D1.7.1 RAM16K[70]%D1.7.1 RAM16K[71]%D1.7.1 RAM16K[72]%D1.7.1 RAM16K[73]%D1.7.1 RAM16K[74]%D1.7.1 RAM16K[75]%D1.7.1 RAM16K[76]%D1.7.1 RAM16K[77]%D1.7.1 RAM16K[78]%D1.7.1 RAM16K[79]%D1.7.1 RAM16K[80]%D1.7.1 RAM16K[81]%D1.7.1 RAM16K[82]%D1.7.1 RAM16K[83]%D1.7.1 RAM16K[84]%D1.7.1 RAM16K[85]%D1.7.1 RAM16K[86]%D1.7.1 RAM16K[87]%D1.7.1 RAM16K[88]%D1.7.1 RAM16K[89]%D1.7.1 RAM16K[90]%D1.7.1 RAM16K[91]%D1.7.1 RAM16K[92]%D1.7.1 RAM16K[93]%D1.7.1 RAM16K[94]%D1.7.1 RAM16K[95]%D1.7.1 RAM16K[96]%D1.7.1 RAM16K[97]%D1.7.1 RAM16K[98]%D1.7.1 RAM16K[99]%D1.7.1 RAM16K[100]%D1.7.1 RAM16K[101]%D1.7.1 RAM16K[102]%D1.7.1 RAM16K[103]%D1.7.1 RAM16K[104]%D1.7.1 RAM16K[105]%D1.7.1 RAM16K[106]%D1.7.1 RAM16K[107]%D1.7.1 RAM16K[108]%D1.7.1 RAM16K[109]%D1.7.1 RAM16K[110]%D1.7.1 RAM16K[111]%D1.7.1 RAM16K[112]%D1.7.1 RAM16K[113]%D1.7.1 RAM16K[114]%D1.7.1 RAM16K[115]%D1.7.1 RAM16K[116]%D1.7.1 RAM16K[117]%D1.7.1 RAM16K[118]%D1.7.1 RAM16K[119]%D1.7.1 RAM16K[120]%D1.7.1 RAM16K[121]%D1.7.1 RAM16K[122]%D1.7.1 RAM16K[123]%D1.7.1 RAM16K[124]%D1.7.1 RAM16K[125]%D1.7.1 RAM16K[126]%D1.7.1 RAM16K[127]%D1.7.1 RAM16K[128]%D1.7.1 RAM16K[129]%D1.7.1 RAM16K[130]%D1.7.1 RAM16K[131]%D1.7.1 RAM16K[132]%D1.7.1 RAM16K[133]%D1.7.1 RAM16K[134]%D1.7.1 RAM16K[135]%D1.7.1 RAM16K[136]%D1.7.1 RAM16K[137]%D1.7.1 RAM16K[138]%D1.7.1 RAM16K[139]%D1.7.1 RAM16K[140]%D1.7.1 RAM16K[141]%D1.7.1 RAM16K[142]%D1.7.1 RAM16K[143]%D1.7.1 RAM16K[144]%D1.7.1 RAM16K[145]%D1.7.1 RAM16K[146]%D1.7.1 RAM16K[147]%D1.7.1 RAM16K[148]%D1.7.1 RAM16K[149]%D1.7.1 RAM16K[150]%D1.7.1 RAM16K[151]%D1.7.1 RAM16K[152]%D1.7.1 RAM16K[153]%D1.7.1 RAM16K[154]%D1.7.1 RAM16K[155]%D1.7.1 RAM16K[156]%D1.7.1 RAM16K[157]%D1.7.1 RAM16K[158]%D1.7.1 RAM16K[159]%D1.7.1 RAM16K[160]%D1.7.1 RAM16K[161]%D1.7.1 RAM16K[162]%D1.7.1 RAM16K[163]%D1.7.1 RAM16K[164]%D1.7.1 RAM16K[165]%D1.7.1 RAM16K[166]%D1.7.1 RAM16K[167]%D1.7.1 RAM16K[168]%D1.7.1 RAM16K[169]%D1.7.1 RAM16K[170]%D1.7.1 RAM16K[171]%D1.7.1 RAM16K[172]%D1.7.1 RAM16K[173]%D1.7.1 RAM16K[174]%D1.7.1 RAM16K[175]%D1.7.1 RAM16K[176]%D1.7.1 RAM16K[177]%D1.7.1 RAM16K[178]%D1.7.1 RAM16K[179]%D1.7.1 RAM16K[180]%D1.7.1 RAM16K[181]%D1.7.1 RAM16K[182]%D1.7.1 RAM16K[183]%D1.7.1 RAM16K[184]%D1.7.1 RAM16K[185]%D1.7.1 RAM16K[186]%D1.7.1 RAM16K[187]%D1.7.1 RAM16K[188]%D1.7.1 RAM16K[189]%D1.7.1 RAM16K[190]%D1.7.1 RAM16K[191]%D1.7.1 RAM16K[192]%D1.7.1 RAM16K[193]%D1.7.1 RAM16K[194]%D1.7.1 RAM16K[195]%D1.7.1 RAM16K[196]%D1.7.1 RAM16K[197]%D1.7.1 RAM16K[198]%D1.7.1 RAM16K[199]%D1.7.1 RAM16K[200]%D1.7.1 RAM16K[201]%D1.7.1 RAM16K[202]%D1.7.1 RAM16K[203]%D1.7.1 RAM16K[204]%D1.7.1 RAM16K[205]%D1.7.1 RAM16K[206]%D1.7.1 RAM16K[207]%D1.7.1 RAM16K[208]%D1.7.1 RAM16K[209]%D1.7.1 RAM16K[210]%D1.7.1 RAM16K[211]%D1.7.1 RAM16K[212]%D1.7.1 RAM16K[213]%D1.7.1 RAM16K[214]%D1.7.1 RAM16K[215]%D1.7.1 RAM16K[216]%D1.7.1 RAM16K[217]%D1.7.1 RAM16K[218]%D1.7.1 RAM16K[219]%D1.7.1 RAM16K[220]%D1.7.1 RAM16K[221]%D1.7.1 RAM16K[222]%D1.7.1 RAM16K[223]%D1.7.1 RAM16K[224]%D1.7.1 RAM16K[225]%D1.7.1 RAM16K[226]%D1.7.1 RAM16K[227]%D1.7.1 RAM16K[228]%D1.7.1 RAM16K[229]%D1.7.1 RAM16K[230]%D1.7.1 RAM16K[231]%D1.7.1 RAM16K[232]%D1.7.1 RAM16K[233]%D1.7.1 RAM16K[234]%D1.7.1 RAM16K[235]%D1.7.1 RAM16K[236]%D1.7.1 RAM16K[237]%D1.7.1 RAM16K[238]%D1.7.1 RAM16K[239]%D1.7.1 RAM16K[240]%D1.7.1 RAM16K[241]%D1.7.1 RAM16K[242]%D1.7.1 RAM16K[243]%D1.7.1 RAM16K[244]%D1.7.1 RAM16K[245]%D1.7.1 RAM16K[246]%D1.7.1 RAM16K[247]%D1.7.1 RAM16K[248]%D1.7.1 RAM16K[249]%D1.7.1 RAM16K[250]%D1.7.1 RAM16K[251]%D1.7.1 RAM16K[252]%D1.7.1 RAM16K[253]%D1.7.1 RAM16K[254]%D1.7.1 RAM16K[255]%D1.7.1 RAM16K[256]%D1.7.1 RAM16K[257]%D1.7.1 RAM16K[258]%D1.7.1 RAM16K[259]%D1.7.1 RAM16K[260]%D1.7.1 RAM16K[261]%D1.7.1 RAM16K[262]%D1.7.1 RAM16K[263]%D1.7.1 RAM16K[264]%D1.7.1 RAM16K[265]%D1.7.1 RAM16K[266]%D1.7.1 RAM16K[267]%D1.7.1 RAM16K[268]%D1.7.1 RAM16K[269]%D1.7.1 RAM16K[270]%D1.7.1 RAM16K[271]%D1.7.1 RAM16K[272]%D1.7.1 RAM16K[273]%D1.7.1
```

End of script - Comparison ended successfully

Slow  Fast

Animate: Program flow Format: Decimal View: Script

Chip Name : Computer (Clocked) Time : 25

Input pins

Name	Value
reset	0

Output pins

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

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/Computer.hdl
```

```
/**
 * The Computer chip, i.e. the nand2tetris Computer.
 * The Computer chip consists of a single nand2tetris
 * It is assumed that the ROM is initialized with the
 * The Computer chip has a single output pin, "outM".
 * When reset is 0, the stored program counter is
 * When reset is 1, the program counter is set to
 * Thus, to start a program, the user must set the
 * and "down" (0). From this point, the user can
 */
```

Internal pins

Name	Value
pc[15]	14
instruction[16]	14
memOut[16]	23456
outM[16]	0
writeM	0
addressM[15]	2

```
// 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;
}
```


File View Run Help

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

Input pins		Output pins	
Name	Value	Name	Value
reset	0		

Internal pins	
Name	Value
pc[15]	24
instruction[16]	-5497
memOut[16]	0
outM[16]	0
writeM	0
addressM[15]	23

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/Computer.hdl

/**
 * The Computer chip, i.e. the Tetris game.
 * The Computer chip consists of a single 16K RAM, a 16K ROM,
 * It is assumed that the ROM is initialized with the Tetris program.
 * The Computer chip has a single 16-bit output, outM[16].
 * When reset is 0, the stored program is loaded into the RAM.
 * When reset is 1, the program is executed.
 * Thus, to start a program, the user must set reset to 1.
 * and "down" (0). From this point on, the program is executed.

```


RAM 16K:

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

ROM:

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

ALU

D Input : 0

M/A Input : 23

ALU output : 0

End of script - Comparison ended successfully

Animate: Program flow Format: Decimal View: Script

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

HDL	Internal pins																														
<pre>// 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 Central Processing unit (CPU) * Consists of an ALU and a set of registers. * execute instructions written in the Hack machine language. * In particular, the ALU executes arithmetic and logical * to the Hack machine language. * The D and A in the language refer to the data and address * while M refers to the memory. * The inM input holds the value of the memory at the address * specified by the addressM input.</pre>	<table> <tr><th>Name</th><th>Value</th></tr> <tr><td>Ainstruction</td><td>1</td></tr> <tr><td>Cinstruction</td><td>0</td></tr> <tr><td>ALUtoA</td><td>0</td></tr> <tr><td>ALUout[16]</td><td>1</td></tr> <tr><td>Aregin[16]</td><td>32767</td></tr> <tr><td>loadA</td><td>1</td></tr> <tr><td>Aout[16]</td><td>32767</td></tr> <tr><td>AMout[16]</td><td>11111</td></tr> <tr><td>loadD</td><td>0</td></tr> <tr><td>Dout[16]</td><td>1</td></tr> <tr><td>ZRout</td><td>0</td></tr> <tr><td>NGout</td><td>0</td></tr> <tr><td>jeq</td><td>0</td></tr> <tr><td>...</td><td>...</td></tr> </table>	Name	Value	Ainstruction	1	Cinstruction	0	ALUtoA	0	ALUout[16]	1	Aregin[16]	32767	loadA	1	Aout[16]	32767	AMout[16]	11111	loadD	0	Dout[16]	1	ZRout	0	NGout	0	jeq	0
Name	Value																														
Ainstruction	1																														
Cinstruction	0																														
ALUtoA	0																														
ALUout[16]	1																														
Aregin[16]	32767																														
loadA	1																														
Aout[16]	32767																														
AMout[16]	11111																														
loadD	0																														
Dout[16]	1																														
ZRout	0																														
NGout	0																														
jeq	0																														
...	...																														

```
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 instruction %B0111111111111111, // @32767
set reset 0;
tick, output, tock, output;
```

End of script - Comparison ended successfully



Animate: Program flow Format: Decimal View: Script

Chip Name: CPU (Clocked) Time: 46

Input pins

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

Output pins

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 Central Processing unit (CPU)
 * Consists of an ALU and a set of registers.
 * execute instructions written in the Hack machine language.
 * In particular, the ALU executes D and A instructions.
 * The D and A in the language refer to the data and address
 * while M refers to the memory.
 * The inM input holds the value of the memory at the address
 * specified by the addressM input.

```

Internal pins

Name	Value
Ainstruction	1
Cinstruction	0
ALUtoA	0
ALUout[16]	1
Aregin[16]	32767
loadA	1
Aout[16]	32767
AMout[16]	11111
loadD	0
Dout[16]	1
ZRout	0
NGout	0
jeq	0
slt	0

```
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 instruction %B0111111111111111, // @32767
set reset 0;
tick, output, tock, output;
```

End of script - Comparison ended successfully

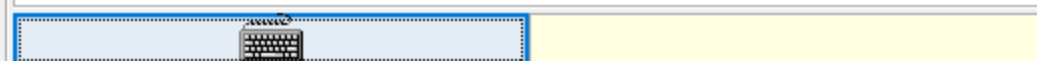
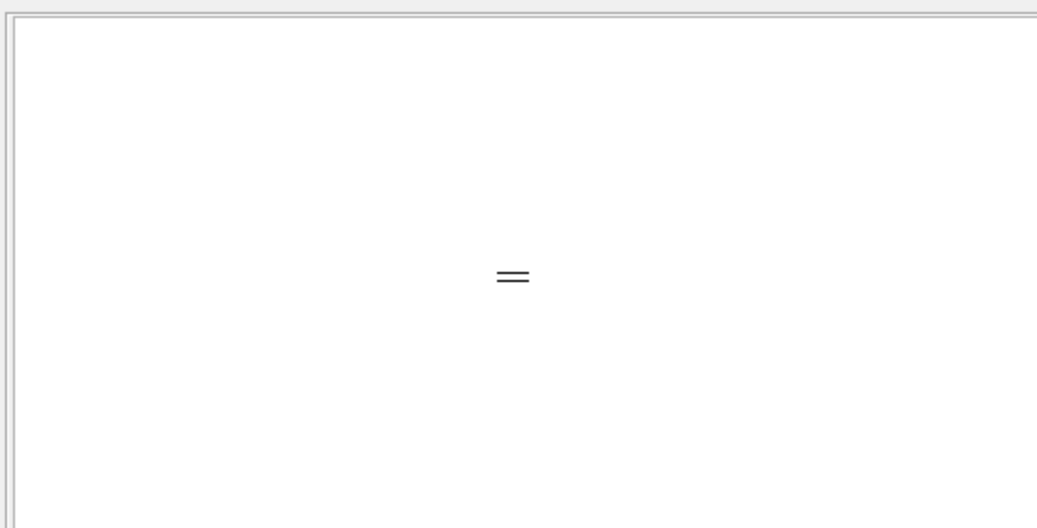


Animate: Program flow Format: Decimal View: Screen

Chip Name : Memory (Clocked) Time : 8

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

HDL	Internal pins																		
<pre>// 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/Memory.hdl /** * This chip implements the computer's data memory, including * The chip facilitates read and write operations. * Read: out(t) = Memory[address(t)] * Write: If load(t-1) then Memory[address(t)] = data(t) * In words: the chip always outputs the data at the memory * location specified by address into the memory location specified by data. */</pre>	<table> <thead> <tr> <th>Name</th><th>Value</th></tr> </thead> <tbody> <tr><td>loadram1</td><td>0</td></tr> <tr><td>loadram2</td><td>0</td></tr> <tr><td>loadscreen</td><td>0</td></tr> <tr><td>loadkbd</td><td>0</td></tr> <tr><td>loadram</td><td>0</td></tr> <tr><td>ramout[16]</td><td>2222</td></tr> <tr><td>scrout[16]</td><td>0</td></tr> <tr><td>kbout[16]</td><td>0</td></tr> </tbody> </table>	Name	Value	loadram1	0	loadram2	0	loadscreen	0	loadkbd	0	loadram	0	ramout[16]	2222	scrout[16]	0	kbout[16]	0
Name	Value																		
loadram1	0																		
loadram2	0																		
loadscreen	0																		
loadkbd	0																		
loadram	0																		
ramout[16]	2222																		
scrout[16]	0																		
kbout[16]	0																		



RAM 16K:		
0189	0	^
0190	0	
0191	0	
0192	2222	
0193	0	
0194	0	
0195	0	v

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 into M?
        addressM[15],     // RAM address (of M)
        pc[15];           // ROM address (of next instruction)

    PARTS:
        // get type of instruction
        Not(in=instruction[15], out=Ainstruction);
        Not(in=Ainstruction, out=Cinstruction);

        And(a=Cinstruction, b=instruction[5], out=ALUtoA); |
        Mux16(a=instruction, b=ALUout, sel=ALUtoA, out=Aregin);

        Or(a=Ainstruction, b=ALUtoA, out=loadA); // load A if A-inst or C-inst&dest to A-reg
        ARegister(in=Aregin, load=loadA, out=Aout);

        Mux16(a=Aout, b=inM, sel=instruction[12], out=AMout); // select A or M based on a-bit

        And(a=Cinstruction, b=instruction[4], out=loadD);
        DRegister(in=ALUout, load=loadD, out=Dout); // load the D register from ALU

        ALU(x=Dout, y=AMout, zx=instruction[11], nx=instruction[10],
            zy=instruction[9], ny=instruction[8], f=instruction[7],
            no=instruction[6], out=ALUout, zr=ZRout, ng=NGout); // calculate

        // Set outputs for writing memory
        Or16(a=false, b=Aout, out[0..14]=addressM);
        Or16(a=false, b=ALUout, out=outM);
        And(a=Cinstruction, b=instruction[3], out=writeM);

        // calc PCload & PCinc - whether to load PC with A reg
        And(a=ZRout, b=instruction[1], out=jeq); // is zero and jump if zero
        And(a=NGout, b=instruction[2], out=jlt); // is neg and jump if neg
        Or(a=ZRout, b=NGout, out=zeroOrNeg);
        Not(in=zeroOrNeg, out=positive); // is positive (not zero and not neg)
        And(a=positive, b=instruction[0], out=jgt); // is pos and jump if pos
        Or(a=jeq, b=jlt, out=jle);
        Or(a=jle, b=jgt, out=jumpToA); // load PC if cond met and jump if cond
        And(a=Cinstruction, b=jumpToA, out=PCload); // Only jump if C instruction
        Not(in=PCload, out=PCinc); // only inc if not load
        PC(in=Aout, inc=PCinc, load=PCload, reset=reset, out[0..14]=pc);

```

}

```
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/05/Memory.hdl
5
6 /**
7  * This chip implements the complete address space of the
8  * computer's data memory, including RAM and memory mapped I/O.
9  * The chip facilitates read and write operations, as follows:
10  *   Read:  out(t) = Memory[address(t)](t)
11  *   Write: If load(t-1) then Memory[address(t-1)](t) = in(t-1)
12  * In words: the chip always outputs the value stored at the memory
13  * location specified by address. If load == 1, the in value is loaded
14  * into the memory location specified by address. This value becomes
15  * available through the out output in the next time step.
16  * Address space rules:
17  * Only the upper 16K+8K+1 words of the Memory chip are used.
18  * Access to address > 0x6000 is invalid. Access to any address in
19  * the range 0x4000 to 0x5FFF results in accessing the screen memory
20  * map. Access to address 0x6000 results in accessing the keyboard
21  * memory map. The behavior in these addresses is described in the
22  * Screen and Keyboard chip specifications given in the book.
23  */
24
25 CHIP Memory {
26     IN in[16], load, address[15];
27     OUT out[16];
28
29     PARTS:
30         DMux4Way(in=load, sel=address[13..14], a=loadram1, b=loadram2, c=loadscreen, d=loadkbd);
31         Or(a=loadram1, b=loadram2, out=loadram);
32         RAM16K(in=in, load=loadram, address=address[0..13], out=ramout);
33         Screen(in=in, load=loadscreen, address=address[0..12], out=scrout);
34         Keyboard(out=kbout);
35         Mux4Way16(a=ramout, b=ramout, c=scrout, d=kbout, sel=address[13..14], out=out);
36 }
37 // 0000 000 RAM start
38 // 0011 FFF RAM end
39 // 0100 000 Screen start
40 // 0101 FFF Screen end
41 // 0110 000 Keyboard
```