

Theory of Computer Sciece: Homework #6 -
NandToTetris — Assembly and Computer

Jeffrey Meyer

November 29, 2018

1 Project 4

Mult Mult.asm

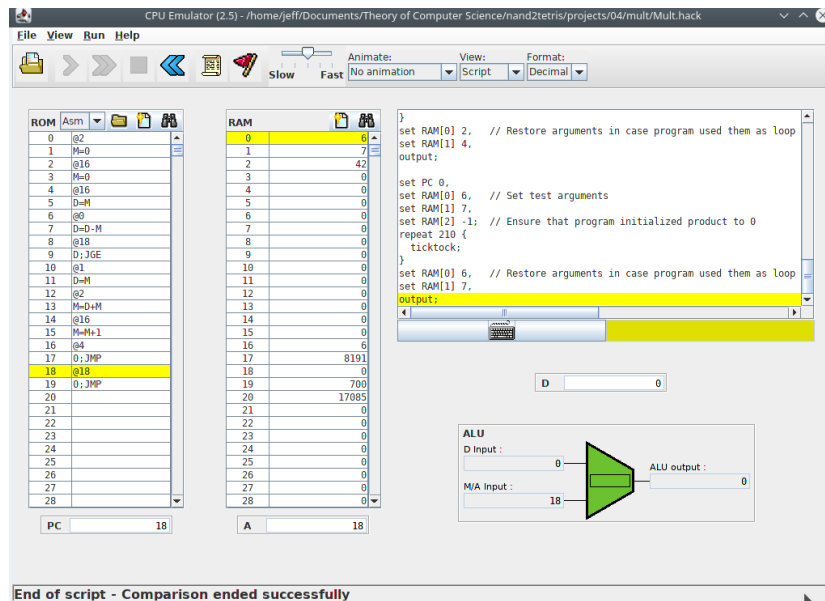
```
// Initialize R2=0
@2
M=0
// Initialize a=0
@a
M=0

(LOOP)
@a
D=M
@0
D=D-M
@HALT
D;JGE // Check if a-r0 < 0

@1
D=M
@2
M=D+M
@a
M=M+1
@LOOP
0;JMP

(HALT)
@HALT
0;JMP
```

All tests passed



Fill Fill.asm

```

@SCREEN
D=A
@SCREEN_END
M=D

// Set screen length and screen end
@8191
D=A
@SCREEN_LENGTH
M=D
@SCREEN_END
M=M+D

// set current position
@SCREEN
D=A

(KEYCHECK)
@KBD
D=M
// If key is pressed FILL_B if not FILL_W
@FILL_B
D;JGT
@FILL_W
0;JMP

```

```

(FILL_B)
    @color
    M=-1
    @FILL
    0;JMP

(FILL_W)
    @color
    M=0
    @FILL
    0;JMP

(FILL)
    @SCREEN_LENGTH
    D=M
    @current
    M=D

(NEXT)
    @current
    D=M
    @position
    M=D
    @SCREEN
    D=A
    @position
    M=M+D

    @color
    D=M
    @position
    A=M
    M=D

    @current
    D=M-1
    M=D

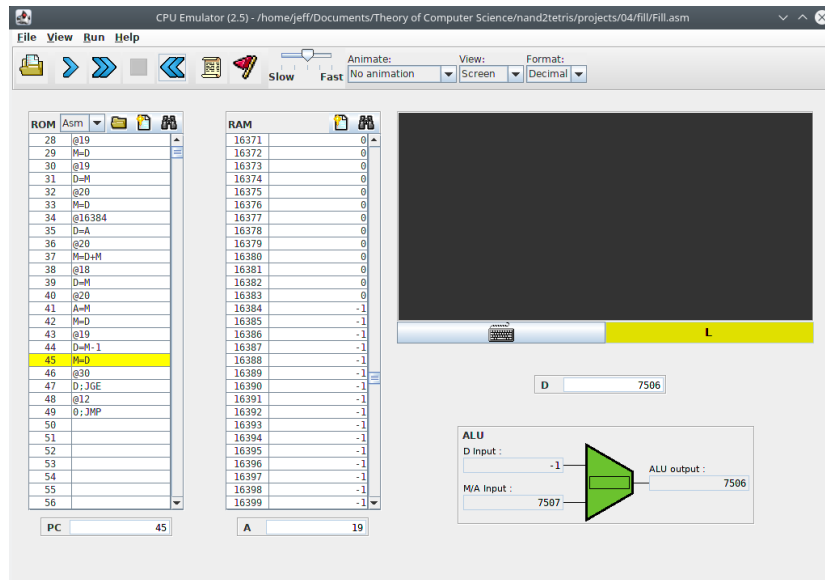
    @NEXT
    D;JGE

@KEYCHECK

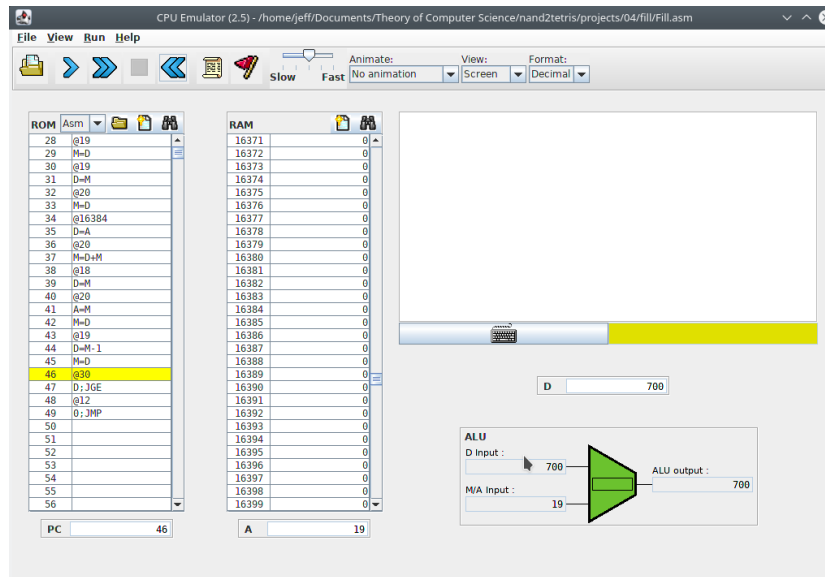
0;JMP

```

Testing here was weird with the keypressing, so I took two screenshots.
 One before and one after pressing a key then pausing.
 First image is pressing *L* while paused.



The second image is after letting off of L after unpausing.



2 Project 5

Memory Memory.hdl

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

PARTS:

```
// DMux on what part to load from
DMux4Way(in=load, sel=address[13..14], a=loadA, b=loadB, c=
    =loadC, d=loadD);
// See if we need to load RAM normally or access keyboard
    or screen
Or(a=loadA, b=loadB, out=loadRAM);
// Define RAM
RAM16K(in=in, load=loadRAM, address=address[0..13], out=
    outRAM);
// Define Screen
Screen(in=in, load=loadC, address=address[0..12], out=
    outScreen);
// Define keyboard
Keyboard(out=outKeyboard);
// Output the output of the selected part
Mux4Way16(sel=address[13..14], a=outRAM, b=outRAM, c=
    outScreen, d=outKeyboard, out=out);
}
```

All tests passed.

Hardware Simulator (2.5) - /home/jeff/Documents/Theory of Computer Science/hand2tetris/projects/05/Memory.hdl

File View Run Help

Chip Na... Memory (Clocked) Time: 8

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

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/Memory.hdl

/**
 * The complete address space of the chip, including RAM and memory-mapped I/O.
 * The chip facilitates read and write access to memory.
 * Read: out(t) = Memory[address(t)]
 * Write: if load(t-1) then Memory[address(t-1)] = in(t-1)
 * In words: the chip always outputs the value stored at the memory location specified by address into the memory location specified by address.
 */

internal pins {
    loadA, loadB, loadC, loadD, loadRAM, outRAM[16], outScreen[16], outKeyboard[16]
}
```

Internal pins	
Name	Value
loadA	0
loadB	0
loadC	0
loadD	0
loadRAM	0
outRAM[16]	2222
outScreen[16]	0
outKeyboard[16]	0

RAM 16K:

8189	0
8190	0
8191	0
8192	2222
8193	0
8194	0
8195	0

out[16] 0

End of script - Comparison ended successfully

CPU CPU.hdl

```
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:
  // =====
  // Instruction Category
  // A = instruction[15] == 0
  // C = instruction[15] == 1

  // Calculate A or C instruction based on first bit
  Not(in=instruction[15], out=aInstruction);
  Not(in=aInstruction, out=cInstruction);

  // =====
  // Destinations
  // M = instruction[3]
  // D = instruction[4]
  // A = instruction[5]

  // Load A
  // loadAfromALU is based on if cInstruction and
  // destination is the A register
  And(a=cInstruction, b=instruction[5], out=loadAfromALU);
  // loadA is based on if the ALU is loading it or the
  // aInstruction is set
  Or(a=aInstruction, b=loadAfromALU, out=loadA);

  // Load D
  And(a=cInstruction, b=instruction[4], out=loadD);

  // Load M
  And(a=cInstruction, b=instruction[3], out=writeM);

  // A input
  Mux16(a=instruction, b=outALU, sel=loadAfromALU, out=
    inputA);

  // =====
  // Operations
  // a-bit = instruction[12] // Whether to use Memory[A] or
  // register A
  Mux16(a=registerA, b=inM, sel=instruction[12], out=AorM);

```

```

// =====
// Jumps
Not(in=zr, out=nzr);
Not(in=ng, out=nng);
And(a=nzr, b=nng, out=pos);

And(a=instruction[0], b=pos, out=jgt);
And(a=instruction[1], b=zr, out=jeq);
And8Way(in=true, in[0]=instruction[0], in[1]=instruction
[1], in[2]=pos, out=jge);
And(a=instruction[2], b=ng, out=jlt);
And8Way(in=true, in[0]=instruction[0], in[1]=instruction
[2], in[2]=nzr, out=jne);
And8Way(in=true, in[0]=instruction[1], in[1]=instruction
[2], in[2]=ng, out=jle);
And8Way(in=true, in[0]=instruction[0], in[1]=instruction
[1], in[2]=instruction[2], out=jmp);

// If unconditional jump, less than or equal to, or
// greather than equal jump
Or8Way(in=false, in[0]=jgt, in[1]=jeq, in[2]=jge, in[3]=
jlt, in[4]=jne, in[5]=jle, in[6]=jmp, out=couldJump);
And(a=cInstruction, b=couldJump, out=jump, out=loadPC);

// =====
// Define parts
ARegister(in=inputA, load=loadA, out=registerA);
DRegister(in=outALU, load=loadD, out=registerD);

ALU(
    x=registerD,
    y=AorM,
    zx=instruction[11],
    nx=instruction[10],
    zy=instruction[9],
    ny=instruction[8],
    f=instruction[7],
    no=instruction[6],
    out=outALU,
    out=outM,
    zr=zr,
    ng=ng
);

// Finish Outputs
Not(in=loadPC, out=incPC);
PC(in=registerA, load=jump, inc=incPC, reset=reset, out
[0..14]=pc);

```



```

    Or16(a=registerA, b=false, out[0..14]=addressM);
}

```

And8Way.hdl

```

CHIP And8Way {
    IN in[8];
    OUT out;

    PARTS:
        And(a=in[0], b=in[1], out=a);
        And(a=a, b=in[2], out=b);
        And(a=b, b=in[3], out=c);
        And(a=c, b=in[4], out=d);
        And(a=d, b=in[5], out=e);
        And(a=e, b=in[6], out=f);
        And(a=f, b=in[7], out=out);
}

```

All tests passed.

CPU.tst

Hardware Simulator (2.5) - /home/jeff/Documents/Theory of Computer Science/nand2tetris/projects/05/CPU.hdl

File View Run Help

Chip Na... 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

```

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

```

Internal pins

Name	Value
aInstruction	1
cInstruction	0
loadAfromALU	0
loadA	1
outALU[16]	0
inputA[16]	32767
registerA[16]	32767
AorM[16]	11111
zr	0
nZr	1
ng	0
nng	1
pos	1

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;

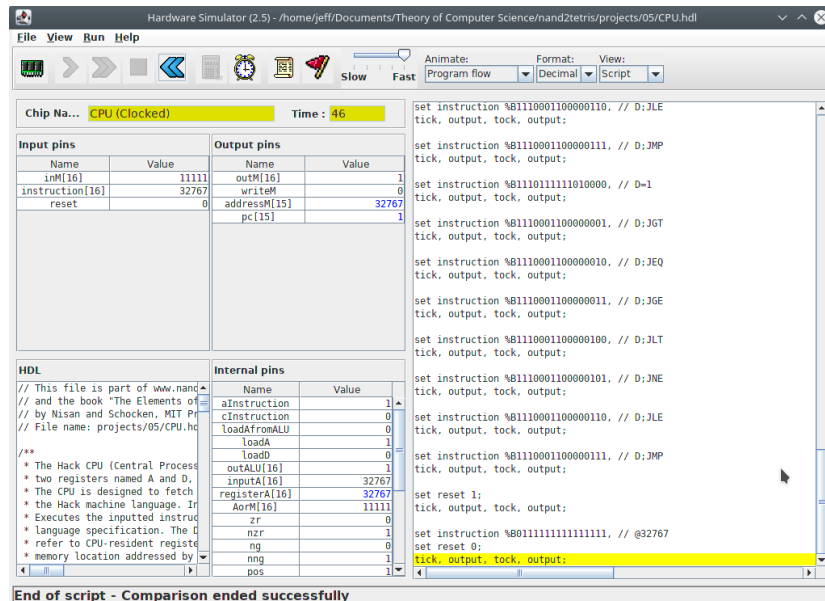
set instruction %B1110001100000111, // D:JMP
tick, output, tock, output;

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

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

End of script - Comparison ended successfully

CPU-External.tst

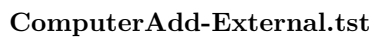


Computer Computer.hdl

```
CHIP Computer {
    IN reset;

    PARTS:
        ROM32K(address=pc, out=instruction);
        CPU(inM=memory, instruction=instruction, reset=reset, outM
            =outM, writeM=writeM, addressM=addressM, pc=pc);
        Memory(in=outM, load=writeM, address=addressM, out=memory)
        ;
}
```

ComputerAdd.tst



Hardware Simulator (2.5) - /home/jeff/Documents/Theory of Computer Science/nand2tetr/projects/05/Computer.hdl

File View Run Help

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

```
// This file is part of www.nand2tetr.com
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press
// File name: projects/05/Computer.hdl

/**
 * The HACK computer, including
 * When reset is 0, the program
 * When reset is 1, the execution
 * Thus, to start a program's execution,
 * and "down" (0). From this point on,
 * the software. In particular,
 * screen may show some output
 * with the computer via the keyboard.
 */

// Internal pins
// Name Value
// pc[15] 14
// instruction[16] 14
// memory[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%$2.1.2 ARegister[]$N01.7.1 DRegister[]$N01.7.1

// 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].

ROM$2K 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

ComputerMax-External.tst

Hardware Simulator (2.5) - /home/jeff/Documents/Theory of Computer Science/nand2tetr/projects/05/Computer.hdl

File View Run Help

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

```
// This file is part of www.nand2tetr.com
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press
// File name: projects/05/Computer.hdl

/**
 * The HACK computer, including
 * When reset is 0, the program
 * When reset is 1, the execution
 * Thus, to start a program's execution,
 * and "down" (0). From this point on,
 * the software. In particular,
 * screen may show some output
 * with the computer via the keyboard.
 */

// Internal pins
// Name Value
// pc[15] 14
// instruction[16] 14
// memory[16] 23456
// outM[16] 0
// writeM 0
// addressM[15] 2
```

File name: projects/05/ComputerMax-external.tst

```
load Computer.hdl,
output-file ComputerMax-external.out,
compare-to ComputerMax-external.cmp,
output-list time%$1.4.1 reset%$2.1.2 RAM16K[0]$N01.7.1 RAM16K[1]$N01.7.1

// 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].

ROM$2K 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

ComputerRect.tst

Hardware Simulator (2.5) - /home/jeff/Documents/Theory of Computer Science/nand2tetris/projects/05/Computer.hdl

File View Run Help

Animate: Program flow Format: Decimal View: Script

Slow Fast

Chip Name: Computer (Clocked) Time: 3

Input pins		Output pins	
Name	Value	Name	Value
reset	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/Computer.hdl

/**
 * The HACK computer, including
 * When reset is 0, the program
 * When reset is 1, the execution
 * Thus, to start a program's execution
 * and "down" (0). From this point
 * the software. In particular,
 * screen may show some output
 * with the computer via the keyboard.
 */
```

Internal pins	
Name	Value
pc[15]	3
instruction[16]	-7418
memory[16]	0
outM[16]	4
writeM	0
addressM[15]	23

```
// 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/ComputerRect.tst

load Computer.hdl.
output-file ComputerRect.out.
compare-to ComputerRect.cmp.
output-list time%S1.4.1 ARegister[]%D1.7.1 PC[]%D0.4.0

// Load a program written in the Hack machine language.
// The program draws a rectangle of width 16 pixels and
// length RAM[0] at the top left of the screen.
ROM32K load Rect.hack.

echo "Before you run this script, select the 'Screen' option from the 'View' menu."
echo "A small rectangle should be drawn at the top left of the screen (the top-left corner of the screen)."

// Draws a rectangle 16 pixels wide and 4 pixels long
set RAM16K[0] 4.
output:
repeat 63 {
    tick, tock, output;
}
```

End of script - Comparison ended successfully

ComputerRect-External.tst

Hardware Simulator (2.5) - /home/jeff/Documents/Theory of Computer Science/nand2tetris/projects/05/Computer.hdl

File View Run Help

Animate: Program flow Format: Decimal View: Script

Slow Fast

Chip Name: Computer (Clocked) Time: 63

Input pins		Output pins	
Name	Value	Name	Value
reset	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/ComputerRect-External.tst

load Computer.hdl.
output-file ComputerRect-external.out.
compare-to ComputerRect-external.cmp.
output-list time%S1.4.1:

// Load a program written in the Hack machine language.
// The program draws a rectangle of width 16 pixels and
// length RAM[0] at the top left of the screen.
ROM32K load Rect.hack.

echo "Before you run this script, select the 'Screen' option from the 'View' menu."
echo "A small rectangle should be drawn at the top left of the screen (the top-left corner of the screen)."

// draw a rectangle 16 pixels wide and 4 pixels long
set RAM16K[0] 4.
output:
repeat 63 {
    tick, tock, output;
}
```

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

End of script - Comparison ended successfully