ROLL NO: IMT2021530

NAME: AKASH PERLA

LINK: https://github.com/akash-35/Nand2tetris

REPORT

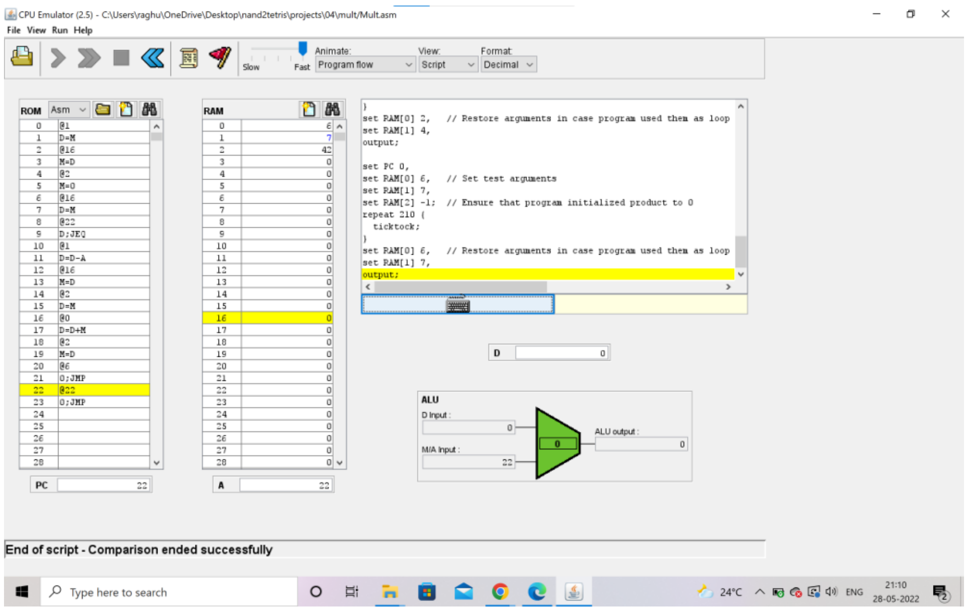
Project 5:

i) Mult.asm

Logic:

i) For multiplying two numbers, we can use repeated addition. For eg: 2x3 can be done as 2+2+2. For performing this, we can use a for loop.
ii) M will store the final result. I will run a loop with i=second number. Then, after performing addition, I will do i=i-1. The loop will be terminated when i=0.

Output:



ROM

```
0   @1
1   D=M
2   @16
3   M=D
4   @2
5   M=0
6   @16
7   D=M
8   @22
9   D;JEQ
10  @1
11  D=D-A
12  @16
13  M=D
14  @2
15  D=M
16  @0
17  D=D+M
18  @2
19  M=D
20  @6
21  0;JMP
22  @22
23  0;JMP
24
25
26
27
28
```

PC 22

RAM

```
0    6
1    7
2   42
3    0
4    0
5    0
6    0
7    0
8    0
9    0
10   0
11   0
12   0
13   0
14   0
15   0
16   0
17   0
18   0
19   0
20   0
21   0
22   0
23   0
24   0
25   0
26   0
27   0
28   0
```

A 22

```
}
set RAM[0] 2,   // Restore arguments in case program used them as loop
set RAM[1] 4,
output;

set PC 0,
set RAM[0] 6,   // Set test arguments
set RAM[1] 7,
set RAM[2] -1;  // Ensure that program initialized product to 0
repeat 210 {
  ticktock;
}
set RAM[0] 6,   // Restore arguments in case program used them as loop
set RAM[1] 7,
output;
```

D 0

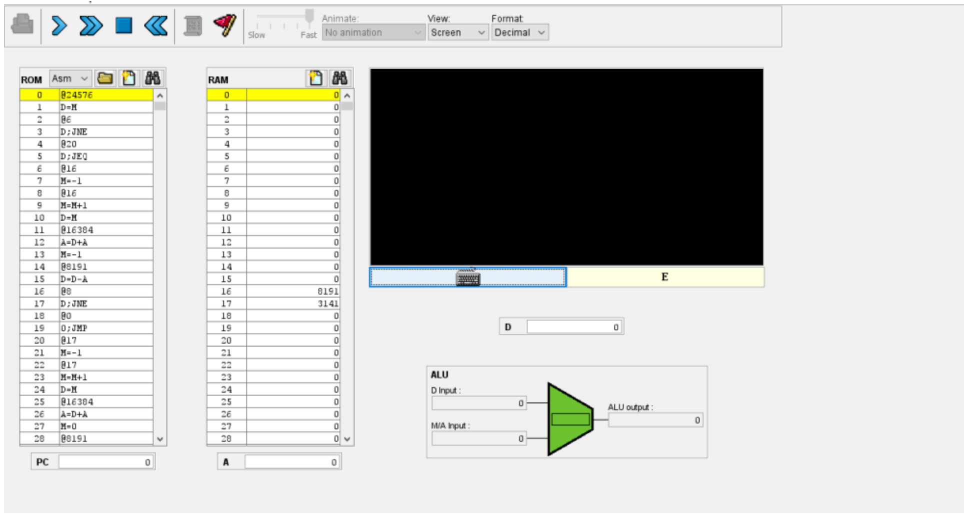
End of script - Comparison ended successfully

ii)Fill.asm

Logic:

- An infinite loop is created. In this loop keyboard input is taken and if key is pressed, the control is shifted to the b label. Else if key is not pressed, the control is shifted to the w Label.
- An inner loop is created to fill all the pixels. The index value is loaded into the A register. Now M[A] is accessed and the value is changed to –1 which denotes black.
- This continues until all the pixels are filled.

Output:



Animate: No animation
View: Screen
Format: Decimal

ROM	Asm	RAM	
0	@24576	0	0
1	D=M	1	0
2	@6	2	0
3	D;JNE	3	0
4	@20	4	0
5	D;JEQ	5	0
6	@16	6	0
7	M=-1	7	0
8	@16	8	0
9	M=M+1	9	0
10	D=M	10	0
11	@16384	11	0
12	A=D+A	12	0
13	M=-1	13	0
14	@8191	14	0
15	D=D-A	15	0
16	@8	16	8191
17	D;JNE	17	3141
18	@0	18	0
19	0;JMP	19	0
20	@17	20	0
21	M=-1	21	0
22	@17	22	0
23	M=M+1	23	0
24	D=M	24	0
25	@16384	25	0
26	A=D+A	26	0
27	M=0	27	0
28	@8191	28	0

PC 0
A 0
E
D 0
D Input : 0
M/A Input : 0
ALU output : 0

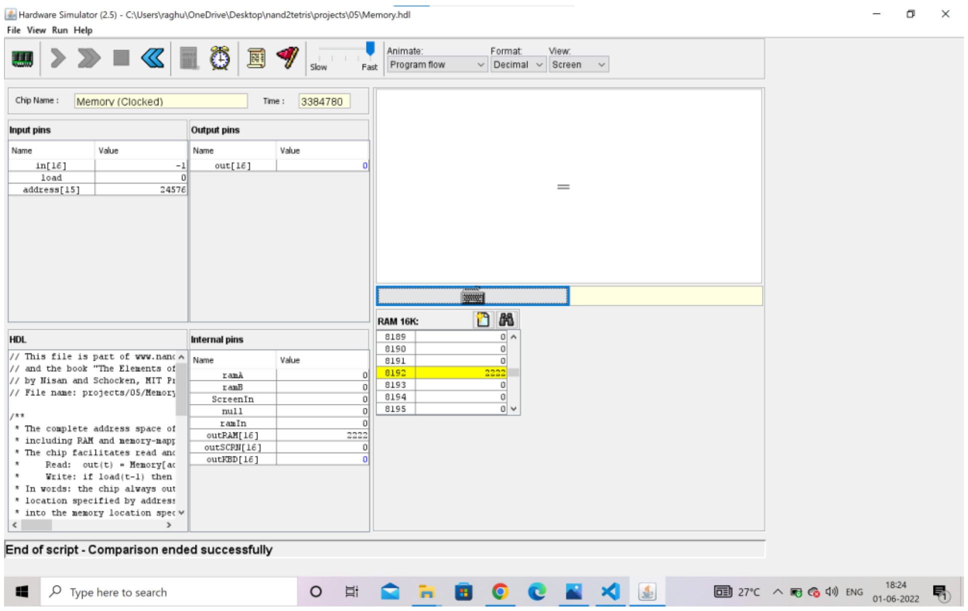
Project 5:

i) Memory.hdl

Logic:

- A 4-way DMux is used in the start to select the path. The select lines are 13 and 14th bit of address.
- An Or gate is used. The output is used as input of RAM16K.
- A 4-way Mux is used at the end to select among the outputs of Keyboard, RAM, OR Screen. The select lines are 13th and 14th bit of address.

Output:



Hardware Simulator (2.5) - C:\Users\raghu\OneDrive\Desktop\nand2tetris\projects\05\Memory.hdl

Animate: Program flow Format: Decimal View: Screen

Chip Name : Memory (Clocked) Time : 3384780

Input pins

Name	Value
in[16]	-1
load	0
address[15]	24576

Output pins

Name	Value
out[16]	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 complete address space of
 * including RAM and memory-mapp
 * The chip facilitates read and
 *     Read:  out(t) = Memory[ac
 *     Write: if load(t-1) then
 * In words: the chip always out
 * location specified by address
 * into the memory location spec
```

Internal pins

Name	Value
ramA	0
ramB	0
ScreenIn	0
null	0
ramIn	0
outRAM[16]	2222
outSCRN[16]	0
outKBD[16]	0

RAM 16K:

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

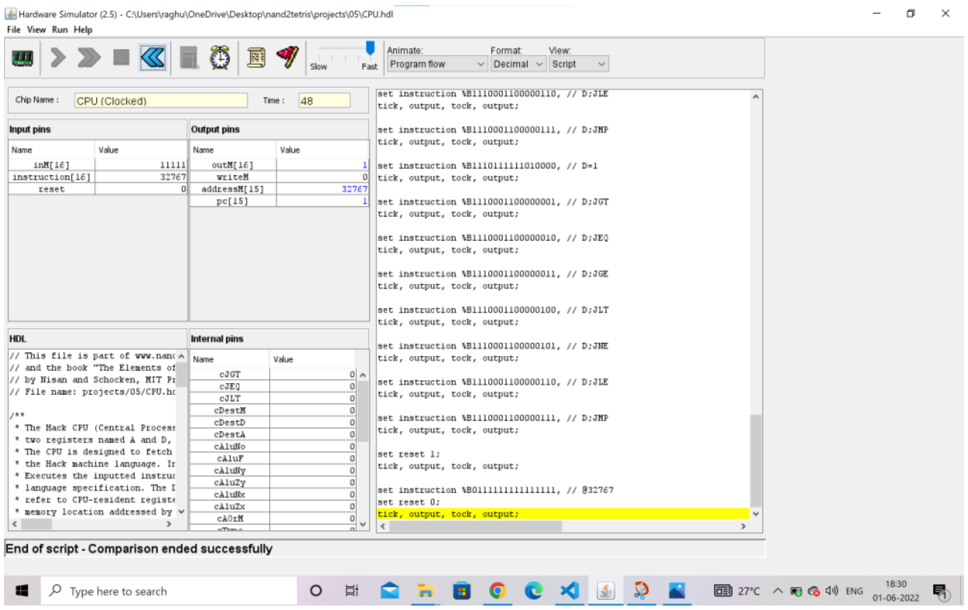
End of script - Comparison ended successfully

ii)CPU.hdl

Logic:

- The CPU requires ALU, MUXs, logic gates, A register, D register.
- A Mux is used to distinguish between A and C type instructions with select line as 15th bit of instruction.
- The internal connections have been made using the schematic diagram provided in the slides

Output:


Hardware Simulator (2.5) - C:\Users\raghu\OneDrive\Desktop\nand2tetris\projects\05\CPU.hdl

Animate: Program flow Format: Decimal View: Script

Chip Name : CPU (Clocked) Time : 48

Input pins

Name	Value
inM[16]	11111
instruction[16]	32767
reset	0

Output pins

Name	Value
outM[16]	1
writeM	0
addressM[15]	32767
pc[15]	1

HDL

```
// This file is part of www.nand
// and the book "The Elements of
// by Nisan and Schocken, MIT Pr
// File name: projects/05/CPU.hd

/**
 * The Hack CPU (Central Process
 * two registers named A and D,
 * The CPU is designed to fetch
 * the Hack machine language. In
 * Executes the inputted instruc
 * language specification. The I
 * refer to CPU-resident registe
 * memory location addressed by
```

Internal pins

Name	Value
cJGT	0
cJEQ	0
cJLT	0
cDestM	0
cDestD	0
cDestA	0
cAluNo	0
cAluF	0
cAluNy	0
cAluZy	0
cAluNx	0
cAluZx	0
cAOrM	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;
```

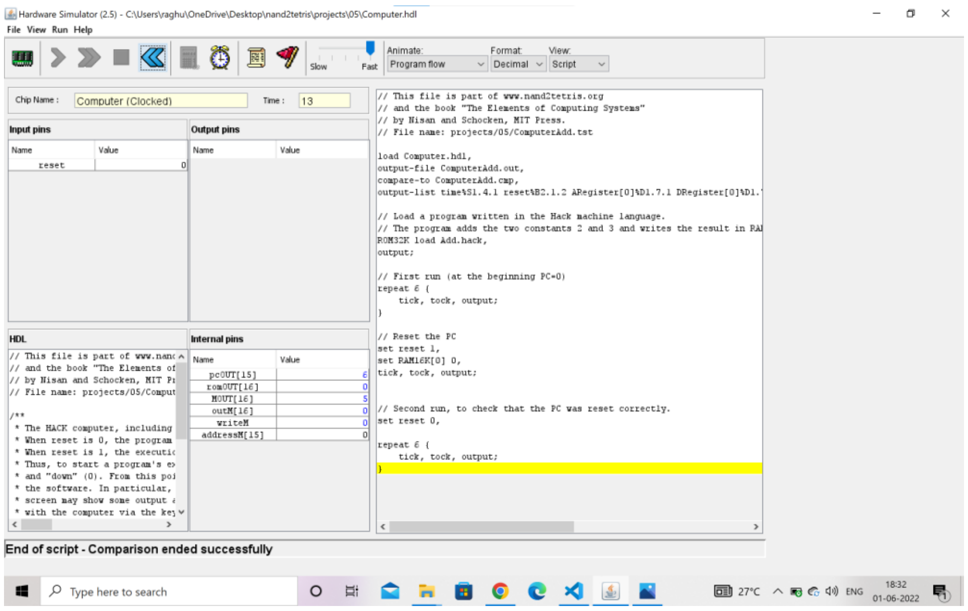
End of script - Comparison ended successfully

iii)Computer.hdl

Logic:

- ROM,CPU and Memory are used.
- For ROM, address is pcOUT,CPU has MOUT has input, romOUT is the instruction.
- For memory, input is outM,load is writeM.

Output:



Hardware Simulator (2.5) - C:\Users\raghu\OneDrive\Desktop\nand2tetris\projects\05\Computer.hdl

Animate: Program flow   Format: Decimal   View: Script

Chip Name : Computer (Clocked)   Time : 13

Input pins

Name	Value
reset	0

Internal pins

Name	Value
pcOUT[15]	6
romOUT[16]	0
MOUT[16]	5
outM[16]	0
writeM	0
addressM[15]	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/Comput

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

```
// 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 RA
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

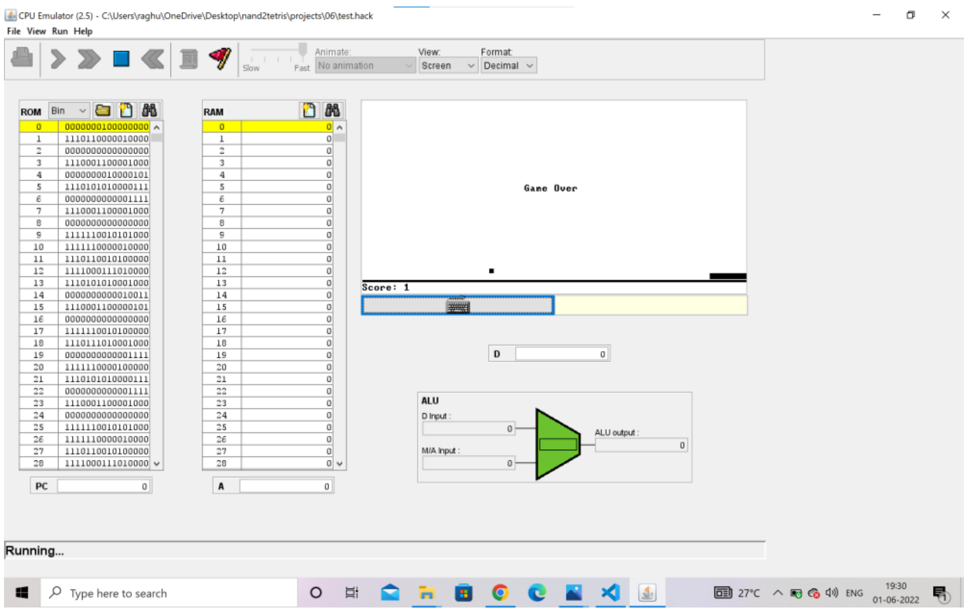
Project 6:

i) Assembler:

Logic:

- By file reading, I will store the contents in an array. I will remove comments, empty lines and spaces.
- The assembly code is generated by using the instructions in the slides.
- For Labels, we remove them from the array, Find the index and convert into binary and add 1 to count.
- Then we write back into test.hack.

Output:



CPU Emulator (2.5) - C:\Users\raghu\OneDrive\Desktop\nand2tetris\projects\06\test.hack

Animate: No animation   View: Screen   Format: Decimal

ROM (Bin)		RAM	
0	0000000100000000	0	0
1	1110110000010000	1	0
2	0000000000000000	2	0
3	1110001100001000	3	0
4	0000000010000101	4	0
5	1110101010000111	5	0
6	0000000000001111	6	0
7	1110001100001000	7	0
8	0000000000000000	8	0
9	1111110010101000	9	0
10	1111110000010000	10	0
11	1110110010100000	11	0
12	1111000111010000	12	0
13	1110101010001000	13	0
14	0000000000010011	14	0
15	1110001100000101	15	0
16	0000000000000000	16	0
17	1111110010100000	17	0
18	1110111010001000	18	0
19	0000000000001111	19	0
20	1111110000100000	20	0
21	1110101010000111	21	0
22	0000000000001111	22	0
23	1110001100001000	23	0
24	0000000000000000	24	0
25	1111110010101000	25	0
26	1111110000010000	26	0
27	1110110010100000	27	0
28	1111000111010000	28	0

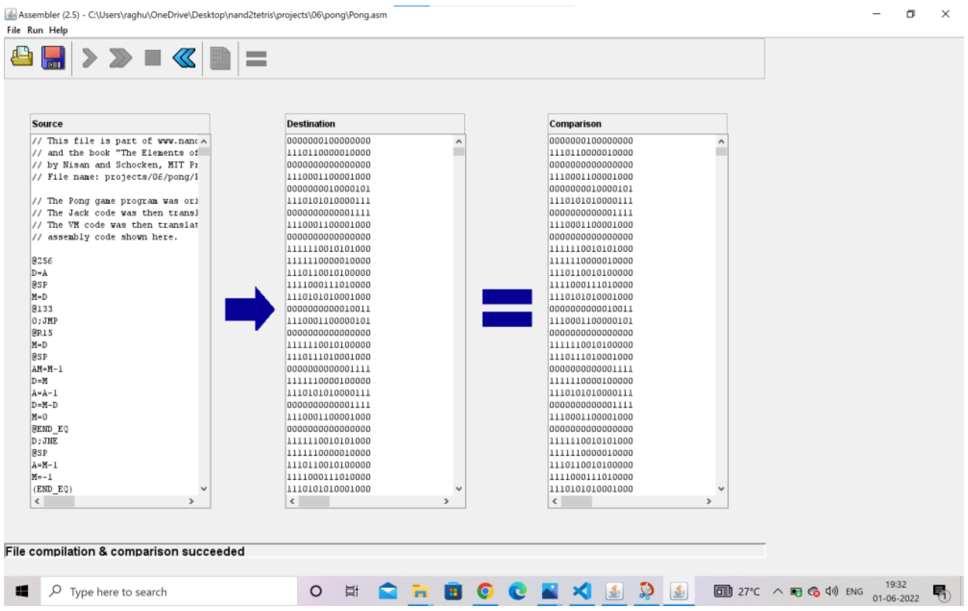
PC 0   A 0   D 0

Game Over

Score: 1

ALU: D Input: 0   M/A Input: 0   ALU output: 0

Running...


Assembler (2.5) - C:\Users\raghu\OneDrive\Desktop\nand2tetris\projects\06\pong\Pong.asm
File Run Help

Source

```
// This file is part of www.nanc
// and the book "The Elements of
// by Nisan and Schocken, MIT Pr
// File name: projects/06/pong/I

// The Pong game program was ori
// The Jack code was then transl
// The VM code was then translat
// assembly code shown here.

@256
D=A
@SP
M=D
@133
0;JMP
@R15
M=D
@SP
AM=M-1
D=M
A=A-1
D=M-D
M=0
@END_EQ
D;JNE
@SP
A=M-1
M=-1
(END_EQ)
```

Destination

```
0000000100000000
1110110000010000
0000000000000000
1110001100001000
0000000010000101
1110101010000111
0000000000001111
1110001100001000
0000000000000000
1111110010101000
1111110000010000
1110110010100000
1111000111010000
1110101010001000
0000000000010011
1110001100000101
0000000000000000
1111110010100000
1110111010001000
0000000000001111
1111110000100000
1110101010000111
0000000000001111
1110001100001000
0000000000000000
1111110010101000
1111110000010000
1110110010100000
1111000111010000
1110101010001000
```

Comparison

```
0000000100000000
1110110000010000
0000000000000000
1110001100001000
0000000010000101
1110101010000111
0000000000001111
1110001100001000
0000000000000000
1111110010101000
1111110000010000
1110110010100000
1111000111010000
1110101010001000
0000000000010011
1110001100000101
0000000000000000
1111110010100000
1110111010001000
0000000000001111
1111110000100000
1110101010000111
0000000000001111
1110001100001000
0000000000000000
1111110010101000
1111110000010000
1110110010100000
1111000111010000
1110101010001000
```

File compilation & comparison succeeded