ELEMENTS OF COMPUTING SYSTEMS-II

A Project Report

Submitted by

Group – 17

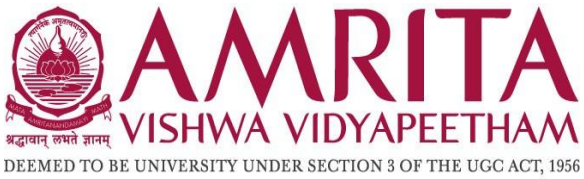
GUNA VARDHAN	(CB.EN.U4AIE22145)
VISWANADH	(CB.EN.U4AIE22105)
PRANAY	(CB.EN.U4AIE22143)
DAKSHINYA	(CB.EN.U4AIE22169)
V SNEGA SRI	(CB.EN.U4AIE22163)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

CSE(AI)



Centre for Computational Engineering and Networking

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

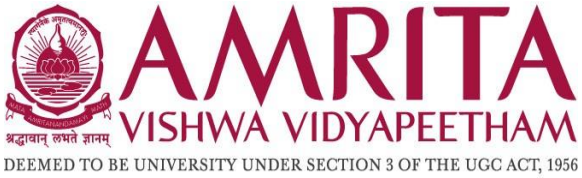
COIMBATORE - 641 112
(INDIA)

JULY - 2023

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112



BONAFIDE CERTIFICATE

This is to certify that the report entitled "ELEMENTS OF COMPUTING SYSTEMS-II" submitted by PULAGAM GUNA VARDHAN(CB.EN.U4AIE22145), ANNA V VISWANADH (CB.EN.U4AIE22105),PAYYAVULA VENKATA PRANAY (CB.EN.U4AIE22143),N DAKSHINYA (CB.EN.U4AIE22169), A SNEGA SREE (CB.EN.U4AIE22163) for the award of the Degree of Bachelor of Technology in the "CSE(AI) " is a bonafide record of the work carried out by her under our guidance and supervision at Amrita School of Artificial Intelligence, Coimbatore.

Ms. Sreelakshmi K

Project Guide

Dr. K.P.Soman

Professor and Head CEN

Submitted for the university examination held on 13-07-2023.

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE
AMRITA VISHWA VIDYAPEETHAM
COIMBATORE - 641 112

DECLARATION

We,PULAGAM GUNA VARDHAN(CB.EN.U4AIE22145), ANNA V VISWANADH (CB.EN.U4AIE22105), PAYYAVULA VENKATA PRANAY (CB.EN.U4AIE22143), N DAKSHINYA (CB.EN.U4AIE22169),A SNEGA SREE(CB.EN.U4AIE22163) hereby declare that this report entitled "ELEMENTS OF COMPUTING SYSTEMS-II", is the record of the original work done by me under the guidance of Ms. Sreelakshmi K, Assistant Professor, Centre for Computational Engineering and Networking, Amrita School of Artificial Intelligence, Coimbatore. To the best of my knowledge this work has not formed the basis for the award of any degree/diploma/ associate ship/fellowship/or a similar award to any candidate in any University.

Place: Coimbatore
Date: 02-07-2023

Signature of the the Student

List of Contents:

ACKNOWLEDGEMENT 6
ABSTRACT 7
1. PART – A 8
1.1 INTRODUCTION 8
1.2 METHODOLOGY 8
1.2.1 ROM32K 9
1.2.2 CPU 9
1.2.3 ALU 11
1.2.4 PC 13
1.2.5 REGISTER 27
1.2.6 MUX 16 28
1.2.7 OR 16 29
1.2.8 NOT GATE 29
1.2.9 AND GATE 31
1.2.10 OR GATE 31
1.2.11 MEMORY 32
1.2.12 RAM16K 33
1.2.13 DMUX4WAY 34
1.2.14 DMUX4WAY16 34
1.3 EXPERIMENT AND RESULTS 35
1.3.1 HDL CODE 35
1.3.2 TST FILE 36
1.4 OUTPUT 37
2. PART B 38
2.1 INTRODUCTION 38
2.2 METHODOLOGY 39
2.2.1 Ball 39
2.2.2 Game 41
2.2.3 Player 49
2.2.4 Main 54
3. PART C 58
3.1INTRODUCTION 58

3.2METHODOLOGY 59
3.2.1ClearScreen 59
3.2.2 RandomNumberGenerator 61
3.2.3 Main 64
3.3EXPERIMENTS & RESULTS: 82
CONCLUSION 86

List of Figures :

Figure 1 HACK CPU 9
Figure 2 ALU 11
Figure 3 Simple Not Gate 30
Figure 4 Simple And Gate 31
Figure 5 Simple Or Gate 32
Figure 6 Part A output-1 37
Figure 7 Part A output-2 37
Figure 8 Part B output-1 56
Figure 9 Part B output-2 57
Figure 10 Part B output-3 58
Figure 11 Part B output-4 58
Figure 12 Part C output-1 82
Figure 13 Part C output-2 83
Figure 14 Part C output-3 83
Figure 15 Part C output-4 84
Figure 16 Part C output-5 84
Figure 17 Part C output-6 85

ACKNOWLEDGEMENT

The successful completion of this project would not have been possible without the support and guidance of several individuals and organizations. We express our sincere gratitude to AMRITA VISHWA VIDYAPEETHAM, who provided us with the resources and facilities necessary to carry out this project.

We would also like to extend our appreciation to our advisor, Ms. Sreelakshmi ma'am, Assistant professor who provided invaluable guidance and support throughout the project.

Additionally, we would like to acknowledge the efforts of our project team members, who worked tirelessly to bring this project to fruition. Their hard work and dedication were critical to our success.

Finally, we would like to thank all the stakeholders who provided support in various ways. Your contributions have been instrumental in making this project a reality.

We are truly grateful for all the support received, and we hope that this project will serve as a testament to the positive impact of collaboration and teamwork.

ABSTRACT

PART – A

Implementing the HACK computer in HDL (Hardware Description Language) and testing its operation using a provided.hack file are the assigned tasks. The HACK computer's architecture, including the CPU, RAM, and input/output devices, is specified in the computer.hdl file, as mentioned in the Nand2Tetris Project 05. Studying the specs, creating the computer.hdl file, and using a hardware simulator, such the one included in the Nand2Tetris software suite, to load and test the computer using a particular.hack file are all necessary steps in the implementation process. The objective is to confirm that the HACK computer is functioning properly and carrying out the intended commands as given in the.hack file.

PART – B

The task is to implement a Pong game using the JACK programming language. Pong is a classic arcade game where players control a paddle to hit a ball back and forth across a screen, aiming to score points against their opponent. The implementation will involve designing and coding the game logic, including the movement of the paddle and ball, collision detection, scoring system, and user input handling. The game will be developed using the JACK language and will utilize various JACK libraries for graphics rendering, input/output, and screen manipulation. The goal is to create a functional and enjoyable Pong game that can be played by users, providing a visual display of the game elements, accurate collision physics, and a responsive user interface.

PART - C

The goal is to use the JACK programming language to construct a hand cricket game. In the common game of hand cricket, participants alternate between batting and bowling with the goal of scoring runs and eliminating the opposition. Designing and implementing the game logic, which includes processing player input, creating random numbers for bowling, a scoring system, and win/lose conditions, are all required for implementation. The game will be created in the JACK programming language, and its user input/output, random number generation, and score tracking will all be supported by various JACK libraries. The objective is to develop a playable and engaging hand cricket game that lets players select their batting gestures and compete against the bowling gestures of

the computer. The game will have a simple the user interface, accurate scoring.

1. PART – A

1.1 INTRODUCTION

The Nand2Tetris course provides a thorough examination of computer engineering and electronics design while assisting students in building the Hack computer, a fully functional general-purpose computer. Participants create an operating system that can execute intricate programmes like Tetris by utilising the fundamental NAND logic gate. The course was painstakingly created by Noam Nisan and Simon Schocken to cover the assembly of various logic gates, combinational chips, and the arithmetic logic unit (ALU). Additionally, Data Flip Flops (DFF) are used as the fundamental building block in the construction of sequential logic circuits, which are essential for the memory in the Hack computer. The implementation of crucial elements including Bits, Registers, RAM chips, and a Programme Counter enables students to have a hands-on understanding of how hardware chips operate.

1.2 METHODOLOGY

The ROM32K, CPU, and Memory make up the three primary parts of the Hack Computer. The initial programme instructions are stored in the built-in component known as ROM32K. Different logic gates, including Not, And, Or, Or16, Mux16, ALU, Register, and PC, are used to build the CPU. The RAM16K, DMux4Way, RAM16K, Screen, Keyboard, and Mux4Way16 components make up the Memory component. Once built, the CPU and memory are combined to create the whole Hack Computer system.

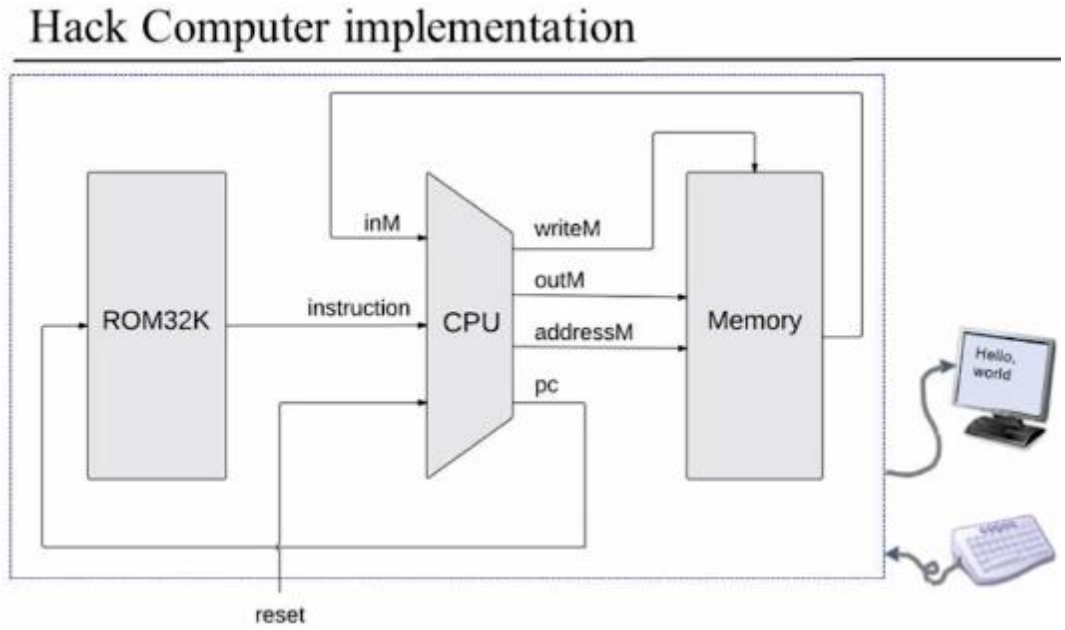


Figure 1 HACK CPU

1.2.1 ROM32K

The Hack Computer makes use of an internal read-only memory (ROM) that has 16K registers with a width of 16 bits per register. The value stored at the specified memory address is always output by this ROM chip, which is a crucial component of the nand2tetris download.

1.2.2 CPU

The Hack CPU is made to retrieve and carry out Hack machine language instructions. It must be able to carry out both the A and C types of instructions that are commonly utilised. While C instructions carry out calculations and store the results in various locations, A instructions load the supplied values.

Not, And, Or, Register, ALU, Or16, Mux16, and PC (Programme Counter) chips are used to construct the CPU.

```
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=ni);
       Mux16(a=outtM,b=instruction,sel=ni,out=i);

       Or(a=ni,b=instruction[5],out=intoA);
       ARegister(in=i,load=intoA,out=A,out[0..14]=addressM);

       And(a=instruction[15],b=instruction[12],out=AorM);
       Mux16(a=A,b=inM,sel=AorM,out=AM);

       ALU(x=D,y=AM,zx=instruction[11],nx=instruction[10],zy=instruction[9],
ny=instruction[8],f=instruction[7],no=instruction[6],out=outtM,out=outM,zr=zr,n
g=ng);

       And(a=instruction[15],b=instruction[4],out=intoD);
       DRegister(in=outtM,load=intoD,out=D);

       And(a=instruction[15],b=instruction[3],out=writeM);

       Not(in=ng,out=pos);
       Not(in=zr,out=nzr);
       And(a=instruction[15],b=instruction[0],out=jgt);
       And(a=pos,b=nzr,out=posnzr);
       And(a=jgt,b=posnzr,out=ld1);
```

```
        And(a=instruction[15],b=instruction[1],out=jeq);
        And(a=jeq,b=zr,out=ld2);

        And(a=instruction[15],b=instruction[2],out=jlt);
        And(a=jlt,b=ng,out=ld3);

        Or(a=ld1,b=ld2,out=ldt);
        Or(a=ld3,b=ldt,out=ld);

        PC(in=A,load=ld,inc=true,reset=reset,out[0..14]=pc);
}
```

1.2.3 ALU

Using two inputs and a number of pre-defined arithmetic and logical functions, the ALU (Arithmetic Logical Unit) computes one function and outputs the result. The 16-bit output from the Hack ALU's operation on two 16-bit values. There are just 18 functions in the Hack ALU, and six 1-bit inputs determine which function is computed. Additionally, it features zr and ng, two 1-bit status outputs.

The Hack ALU is capable of carrying out the following 18 operations: 0, 1, -1, x, y,!x,!y, -x, -y, x+1, y+1, x-1, y-1, x+y, x-y, y-x, x&y, and x|y.

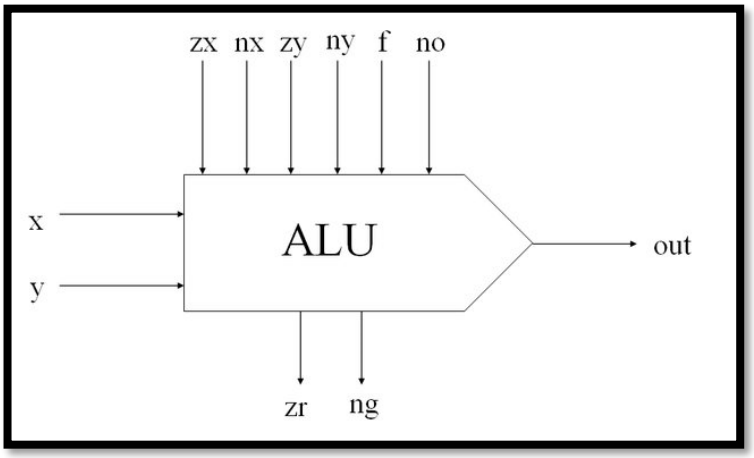


Figure 2 ALU

```
CHIP ALU {
    IN
        x[16], y[16],  // 16-bit inputs
        zx, // zero the x input?
        nx, // negate the x input?
        zy, // zero the y input?
        ny, // negate the y input?
        f,  // compute out = x + y (if 1) or x & y (if 0)
        no; // negate the out output?

    OUT
        out[16], // 16-bit output
        zr, // 1 if (out == 0), 0 otherwise
        ng; // 1 if (out < 0),  0 otherwise

    PARTS:
   // Put you code here:


Mux16(a=x,b=false,sel=zx,out=o1);
Mux16(a=y,b=false,sel=zy,out=o2);
Not16(in=o1,out=o3);
Mux16(a=o1,b=o3,sel=nx,out=o4);
Not16(in=o2,out=o5);
Mux16(a=o2,b=o5,sel=ny,out=o6);
Add16(a=o4,b=o6,out=o7);
And16(a=o4,b=o6,out=o8);
Mux16(a=o8,b=o7,sel=f,out=o9);
Not16(in=o9,out=o10);
Mux16(a=o9,b=o10,sel=no,out=out,out[0..7]=fout1,out[8..15]=fout2,out[15]=ng);
Or8Way(in=fout1,out=o11);
```

```
Or8Way(in=fout2,out=o12);
Or(a=o11,b=o12,out=o13);
Not(in=o13,out=zr);
}
```

1.2.4 PC

The CPU determines which instruction should be fetched and executed next using the Programme Counter (PC) mechanism. It has three possible control operations: Reset (fetch the first instruction), Inc (fetch the next instruction), and Load (fetch instruction n). It also provides the instruction's address.

It consists of a 16-bit register, three 16-bit Muxes, and a 16-bit incrementor (built of half adders).

```
CHIP PC {
   IN in[16],load,inc,reset;
   OUT out[16];

   PARTS:
   // Put your code here:
Nand(a=a1,b=true,out=x1);
Nand(a=a1,b=x1,out=x2);
Nand(a=x1,b=true,out=x3);
Nand(a=x2,b=x3,out=x4);
Nand(a=x4,b=false,out=x5);
Nand(a=x4,b=x5,out=x6);
Nand(a=x5,b=false,out=x7);
Nand(a=x6,b=x7,out=out0);
Nand(a=x5,b=x1,out=y0);

Nand(a=a2,b=false,out=x8);
Nand(a=a2,b=x8,out=x9);
Nand(a=x8,b=false,out=x10);
Nand(a=x9,b=x10,out=x11);
Nand(a=x11,b=y0,out=x12);
```

```
Nand(a=x11,b=x12,out=x13);
Nand(a=x12,b=y0,out=x14);
Nand(a=x13,b=x14,out=out1);
Nand(a=x12,b=x8,out=y1);

Nand(a=a3,b=false,out=x15);
Nand(a=a3,b=x15,out=x16);
Nand(a=x15,b=false,out=x17);
Nand(a=x16,b=x17,out=x18);
Nand(a=x18,b=y1,out=x19);
Nand(a=x18,b=x19,out=x20);
Nand(a=x19,b=y1,out=x21);
Nand(a=x20,b=x21,out=out2);
Nand(a=x19,b=x15,out=y2);

Nand(a=a4,b=false,out=x22);
Nand(a=a4,b=x22,out=x23);
Nand(a=x22,b=false,out=x24);
Nand(a=x23,b=x24,out=x25);
Nand(a=x25,b=y2,out=x26);
Nand(a=x25,b=x26,out=x27);
Nand(a=x26,b=y2,out=x28);
Nand(a=x27,b=x28,out=out3);
Nand(a=x26,b=x22,out=y3);

Nand(a=a5,b=false,out=x29);
Nand(a=a5,b=x29,out=x30);
Nand(a=x29,b=false,out=x31);
Nand(a=x30,b=x31,out=x32);
Nand(a=x32,b=y3,out=x33);
Nand(a=x32,b=x33,out=x34);
Nand(a=x33,b=y3,out=x35);
Nand(a=x34,b=x35,out=out4);
```

```
Nand(a=x33,b=x29,out=y4);

Nand(a=a6,b=false,out=x36);
Nand(a=a6,b=x36,out=x37);
Nand(a=x36,b=false,out=x38);
Nand(a=x37,b=x38,out=x39);
Nand(a=x39,b=y4,out=x40);
Nand(a=x39,b=x40,out=x41);
Nand(a=x40,b=y4,out=x42);
Nand(a=x41,b=x42,out=out5);
Nand(a=x40,b=x36,out=y5);

Nand(a=a7,b=false,out=x43);
Nand(a=a7,b=x43,out=x44);
Nand(a=x43,b=false,out=x45);
Nand(a=x44,b=x45,out=x46);
Nand(a=x46,b=y5,out=x47);
Nand(a=x46,b=x47,out=x48);
Nand(a=x47,b=y5,out=x49);
Nand(a=x48,b=x49,out=out6);
Nand(a=x47,b=x43,out=y6);

Nand(a=a8,b=false,out=x50);
Nand(a=a8,b=x50,out=x51);
Nand(a=x50,b=false,out=x52);
Nand(a=x51,b=x52,out=x53);
Nand(a=x53,b=y6,out=x54);
Nand(a=x53,b=x54,out=x55);
Nand(a=x54,b=y6,out=x56);
Nand(a=x55,b=x56,out=out7);
Nand(a=x54,b=x50,out=y7);

Nand(a=a9,b=false,out=x57);
```

```
Nand(a=a9,b=x57,out=x58);
Nand(a=x57,b=false,out=x59);
Nand(a=x58,b=x59,out=x60);
Nand(a=x60,b=y7,out=x61);
Nand(a=x60,b=x61,out=x62);
Nand(a=x61,b=y7,out=x63);
Nand(a=x62,b=x63,out=out8);
Nand(a=x61,b=x57,out=y8);

Nand(a=a10,b=false,out=x64);
Nand(a=a10,b=x64,out=x65);
Nand(a=x64,b=false,out=x66);
Nand(a=x65,b=x66,out=x67);
Nand(a=x67,b=y8,out=x68);
Nand(a=x67,b=x68,out=x69);
Nand(a=x68,b=y8,out=x70);
Nand(a=x69,b=x70,out=out9);
Nand(a=x68,b=x64,out=y9);

Nand(a=a11,b=false,out=x71);
Nand(a=a11,b=x71,out=x72);
Nand(a=x71,b=false,out=x73);
Nand(a=x72,b=x73,out=x74);
Nand(a=x74,b=y9,out=x75);
Nand(a=x74,b=x75,out=x76);
Nand(a=x75,b=y9,out=x77);
Nand(a=x76,b=x77,out=out10);
Nand(a=x75,b=x71,out=y10);

Nand(a=a12,b=false,out=x78);
Nand(a=a12,b=x78,out=x79);
Nand(a=x78,b=false,out=x80);
Nand(a=x79,b=x80,out=x81);
```

```
Nand(a=x81,b=y10,out=x82);
Nand(a=x81,b=x82,out=x83);
Nand(a=x82,b=y10,out=x84);
Nand(a=x83,b=x84,out=out11);
Nand(a=x82,b=x78,out=y11);

Nand(a=a13,b=false,out=x85);
Nand(a=a13,b=x85,out=x86);
Nand(a=x85,b=false,out=x87);
Nand(a=x86,b=x87,out=x88);
Nand(a=x88,b=y11,out=x89);
Nand(a=x88,b=x89,out=x90);
Nand(a=x89,b=y11,out=x91);
Nand(a=x90,b=x91,out=out12);
Nand(a=x89,b=x85,out=y12);

Nand(a=a14,b=false,out=x92);
Nand(a=a14,b=x92,out=x93);
Nand(a=x92,b=false,out=x94);
Nand(a=x93,b=x94,out=x95);
Nand(a=x95,b=y12,out=x96);
Nand(a=x95,b=x96,out=x97);
Nand(a=x96,b=y12,out=x98);
Nand(a=x97,b=x98,out=out13);
Nand(a=x96,b=x92,out=y13);

Nand(a=a15,b=false,out=x99);
Nand(a=a15,b=x99,out=x100);
Nand(a=x99,b=false,out=x101);
Nand(a=x100,b=x101,out=x102);
Nand(a=x102,b=y13,out=x103);
Nand(a=x102,b=x103,out=x104);
Nand(a=x103,b=y13,out=x105);
```

```
Nand(a=x104,b=x105,out=out14);
Nand(a=x103,b=x99,out=y14);

Nand(a=a16,b=false,out=x106);
Nand(a=a16,b=x106,out=x107);
Nand(a=x106,b=false,out=x108);
Nand(a=x107,b=x108,out=x109);
Nand(a=x109,b=y14,out=x110);
Nand(a=x109,b=x110,out=x111);
Nand(a=x110,b=y14,out=x112);
Nand(a=x111,b=x112,out=out15);
Nand(a=x110,b=x106,out=drop);

///mux16

Nand(a=inc,b=inc,out=c);
Nand(a=c,b=a1,out=l1);
Nand(a=inc,b=out0,out=l2);
Nand(a=l1,b=l2,out=lOut0);

Nand(a=c,b=a2,out=l3);
Nand(a=inc,b=out1,out=l4);
Nand(a=l3,b=l4,out=lOut1);

Nand(a=c,b=a3,out=l5);
Nand(a=inc,b=out2,out=l6);
Nand(a=l5,b=l6,out=lOut2);

Nand(a=c,b=a4,out=l7);
Nand(a=inc,b=out3,out=l8);
Nand(a=l7,b=l8,out=lOut3);

Nand(a=c,b=a5,out=l9);
```

```
Nand(a=inc,b=out4,out=l10);
Nand(a=l9,b=l10,out=lOut4);

Nand(a=c,b=a6,out=l11);
Nand(a=inc,b=out5,out=l12);
Nand(a=l11,b=l12,out=lOut5);

Nand(a=c,b=a7,out=l13);
Nand(a=inc,b=out6,out=l14);
Nand(a=l13,b=l14,out=lOut6);

Nand(a=c,b=a8,out=l15);
Nand(a=inc,b=out7,out=l16);
Nand(a=l15,b=l16,out=lOut7);

Nand(a=c,b=a9,out=l17);
Nand(a=inc,b=out8,out=l18);
Nand(a=l17,b=l18,out=lOut8);

Nand(a=c,b=a10,out=l19);
Nand(a=inc,b=out9,out=l20);
Nand(a=l19,b=l20,out=lOut9);

Nand(a=c,b=a11,out=l21);
Nand(a=inc,b=out10,out=l22);
Nand(a=l21,b=l22,out=lOut10);

Nand(a=c,b=a12,out=l23);
Nand(a=inc,b=out11,out=l24);
Nand(a=l23,b=l24,out=lOut11);

Nand(a=c,b=a13,out=l25);
Nand(a=inc,b=out12,out=l26);
```

```
Nand(a=l25,b=l26,out=lOut12);

Nand(a=c,b=a14,out=l27);
Nand(a=inc,b=out13,out=l28);
Nand(a=l27,b=l28,out=lOut13);

Nand(a=c,b=a15,out=l29);
Nand(a=inc,b=out14,out=l30);
Nand(a=l29,b=l30,out=lOut14);

Nand(a=s,b=a16,out=oo31);
Nand(a=inc,b=out15,out=oo32);
Nand(a=oo31,b=oo32,out=lOut15);
////mux16

Nand(a=load,b=load,out=L);
Nand(a=L,b=lOut0,out=O1);
Nand(a=load,b=in[0],out=O2);
Nand(a=O1,b=O2,out=oo0);

Nand(a=L,b=lOut1,out=O3);
Nand(a=load,b=in[1],out=O4);
Nand(a=O3,b=O4,out=oo1);

Nand(a=L,b=lOut2,out=O5);
Nand(a=load,b=in[2],out=O6);
Nand(a=O5,b=O6,out=oo2);

Nand(a=L,b=lOut3,out=O7);
Nand(a=load,b=in[3],out=O8);
Nand(a=O7,b=O8,out=oo3);

Nand(a=L,b=lOut4,out=O9);
```

```
Nand(a=load,b=in[4],out=O10);
Nand(a=O9,b=O10,out=oo4);

Nand(a=L,b=lOut5,out=O11);
Nand(a=load,b=in[5],out=O12);
Nand(a=O11,b=O12,out=oo5);

Nand(a=L,b=lOut6,out=O13);
Nand(a=load,b=in[6],out=O14);
Nand(a=O13,b=O14,out=oo6);

Nand(a=L,b=lOut7,out=O15);
Nand(a=load,b=in[7],out=O16);
Nand(a=O15,b=O16,out=oo7);

Nand(a=L,b=lOut8,out=O17);
Nand(a=load,b=in[8],out=O18);
Nand(a=O17,b=O18,out=oo8);

Nand(a=L,b=lOut9,out=O19);
Nand(a=load,b=in[9],out=O20);
Nand(a=O19,b=O20,out=oo9);

Nand(a=L,b=lOut10,out=O21);
Nand(a=load,b=in[10],out=O22);
Nand(a=O21,b=O22,out=oo10);

Nand(a=L,b=lOut11,out=O23);
Nand(a=load,b=in[11],out=O24);
Nand(a=O23,b=O24,out=oo11);

Nand(a=L,b=lOut12,out=O25);
Nand(a=load,b=in[12],out=O26);
```

```
Nand(a=O25,b=O26,out=oo12);

Nand(a=L,b=lOut13,out=O27);
Nand(a=load,b=in[13],out=O28);
Nand(a=O27,b=O28,out=oo13);

Nand(a=L,b=lOut14,out=O29);
Nand(a=load,b=in[14],out=O30);
Nand(a=O29,b=O30,out=oo14);

Nand(a=L,b=lOut15,out=O31);
Nand(a=load,b=in[15],out=O32);
Nand(a=O31,b=O32,out=oo15);
///mux16

Nand(a=reset,b=reset,out=d);
Nand(a=d,b=oo0,out=n1);
Nand(a=reset,b=false,out=n2);
Nand(a=n1,b=n2,out=nu0);

Nand(a=d,b=oo1,out=n3);
Nand(a=reset,b=false,out=n4);
Nand(a=n3,b=n4,out=nu1);

Nand(a=d,b=oo2,out=n5);
Nand(a=reset,b=false,out=n6);
Nand(a=n5,b=n6,out=nu2);

Nand(a=d,b=oo3,out=n7);
Nand(a=reset,b=false,out=n8);
Nand(a=n7,b=n8,out=nu3);

Nand(a=d,b=oo4,out=n9);
```

```
Nand(a=reset,b=false,out=n10);
Nand(a=n9,b=n10,out=nu4);

Nand(a=d,b=oo5,out=n11);
Nand(a=reset,b=false,out=n12);
Nand(a=n11,b=n12,out=nu5);

Nand(a=d,b=oo6,out=n13);
Nand(a=reset,b=false,out=n14);
Nand(a=n13,b=n14,out=nu6);

Nand(a=d,b=oo7,out=n15);
Nand(a=reset,b=false,out=n16);
Nand(a=n15,b=n16,out=nu7);

Nand(a=d,b=oo8,out=n17);
Nand(a=reset,b=false,out=n18);
Nand(a=n17,b=n18,out=nu8);

Nand(a=d,b=oo9,out=n19);
Nand(a=reset,b=false,out=n20);
Nand(a=n19,b=n20,out=nu9);

Nand(a=d,b=oo10,out=n21);
Nand(a=reset,b=false,out=n22);
Nand(a=n21,b=n22,out=nu10);

Nand(a=d,b=oo11,out=n23);
Nand(a=reset,b=false,out=n24);
Nand(a=n23,b=n24,out=nu11);

Nand(a=d,b=oo12,out=n25);
Nand(a=reset,b=false,out=n26);
```

```
Nand(a=n25,b=n26,out=nu12);

Nand(a=d,b=oo13,out=n27);
Nand(a=reset,b=false,out=n28);
Nand(a=n27,b=n28,out=nu13);

Nand(a=d,b=oo14,out=n29);
Nand(a=reset,b=false,out=n30);
Nand(a=n29,b=n30,out=nu14);

Nand(a=d,b=oo15,out=n31);
Nand(a=reset,b=false,out=n32);
Nand(a=n31,b=n32,out=nu15);

Nand(a=inc,b=inc,out=oRn1);
Nand(a=load,b=load,out=oRn2);
Nand(a=oRn1,b=oRn2,out=oRnut);

Nand(a=oRnut,b=oRnut,out=n1oR);
Nand(a=reset,b=reset,out=n2oR);
Nand(a=n1oR,b=n2oR,out=nr);
/////register

Nand(a=nr,b=nr,out=s);
Nand(a=s,b=a1,out=f11);
Nand(a=nr,b=nu0,out=f21);
Nand(a=f11,b=f21,out=m1);
DFF(in=m1,out=out[0],out=a1);

Nand(a=s,b=a2,out=f12);
Nand(a=nr,b=nu1,out=f22);
Nand(a=f12,b=f22,out=m2);
DFF(in=m2,out=out[1],out=a2);
```

```
Nand(a=s,b=a3,out=f13);
Nand(a=nr,b=nu2,out=f23);
Nand(a=f13,b=f23,out=m3);
DFF(in=m3,out=out[2],out=a3);

Nand(a=s,b=a4,out=f14);
Nand(a=nr,b=nu3,out=f24);
Nand(a=f14,b=f24,out=m4);
DFF(in=m4,out=out[3],out=a4);

Nand(a=s,b=a5,out=f15);
Nand(a=nr,b=nu4,out=f25);
Nand(a=f15,b=f25,out=m5);
DFF(in=m5,out=out[4],out=a5);

Nand(a=s,b=a6,out=f16);
Nand(a=nr,b=nu5,out=f26);
Nand(a=f16,b=f26,out=m6);
DFF(in=m6,out=out[5],out=a6);

Nand(a=s,b=a7,out=f17);
Nand(a=nr,b=nu6,out=f27);
Nand(a=f17,b=f27,out=m7);
DFF(in=m7,out=out[6],out=a7);

Nand(a=s,b=a8,out=f18);
Nand(a=nr,b=nu7,out=f28);
Nand(a=f18,b=f28,out=m8);
DFF(in=m8,out=out[7],out=a8);

Nand(a=s,b=a9,out=f19);
Nand(a=nr,b=nu8,out=f29);
```

```
Nand(a=f19,b=f29,out=m9);
DFF(in=m9,out=out[8],out=a9);

Nand(a=s,b=a10,out=f110);
Nand(a=nr,b=nu9,out=f210);
Nand(a=f110,b=f210,out=m10);
DFF(in=m10,out=out[9],out=a10);

Nand(a=s,b=a11,out=f111);
Nand(a=nr,b=nu10,out=f211);
Nand(a=f111,b=f211,out=m11);
DFF(in=m11,out=out[10],out=a11);

Nand(a=s,b=a12,out=f112);
Nand(a=nr,b=nu11,out=f212);
Nand(a=f112,b=f212,out=m12);
DFF(in=m12,out=out[11],out=a12);

Nand(a=s,b=a13,out=f113);
Nand(a=nr,b=nu12,out=f213);
Nand(a=f113,b=f213,out=m13);
DFF(in=m13,out=out[12],out=a13);

Nand(a=s,b=a14,out=f114);
Nand(a=nr,b=nu13,out=f214);
Nand(a=f114,b=f214,out=m14);
DFF(in=m14,out=out[13],out=a14);

Nand(a=s,b=a15,out=f115);
Nand(a=nr,b=nu14,out=f215);
Nand(a=f115,b=f215,out=m15);
DFF(in=m15,out=out[14],out=a15);
```

```
Nand(a=s,b=a16,out=f116);
Nand(a=nr,b=nu15,out=f216);
Nand(a=f116,b=f216,out=m16);
DFF(in=m16,out=out[15],out=a16);
}
```

1.2.5 REGISTER

A register is a collection of flip-flops, each of which has a single bit of memory and operates on a common clock. A collection of n flip-flops that can store n bits of binary data make up an n-bit register.

A Mux and a DFF make up a bit.

```
CHIP Register {
   IN in[16], load;
   OUT out[16];

   PARTS:
   // Put your code here:
  Bit(in=in[0],load=load,out=out[0]);
  Bit(in=in[1],load=load,out=out[1]);
  Bit(in=in[2],load=load,out=out[2]);
  Bit(in=in[3],load=load,out=out[3]);
  Bit(in=in[4],load=load,out=out[4]);
  Bit(in=in[5],load=load,out=out[5]);
  Bit(in=in[6],load=load,out=out[6]);
  Bit(in=in[7],load=load,out=out[7]);
  Bit(in=in[8],load=load,out=out[8]);
  Bit(in=in[9],load=load,out=out[9]);
  Bit(in=in[10],load=load,out=out[10]);
  Bit(in=in[11],load=load,out=out[11]);
  Bit(in=in[12],load=load,out=out[12]);
  Bit(in=in[13],load=load,out=out[13]);
```

```
  Bit(in=in[14],load=load,out=out[14]);
  Bit(in=in[15],load=load,out=out[15]);

}
```

1.2.6 MUX 16

A multiplexer, often known as a mux or combinational circuit, is a device that chooses binary data from one of numerous input lines and sends it to a single output line. A group of selection lines determine which input line will be chosen. There are n selection lines and 2n input lines, and these lines determine which input is chosen.

Two inputs, a selection line, and one output make up a 2-to-1 MUX. We are creating a bitwise 2-to-1 MUX called a Mux16 using only Nand gates.

```
CHIP Mux16 {
   IN a[16], b[16], sel;
   OUT out[16];

   PARTS:
   Mux(a=a[0],b=b[0],sel=sel,out=out[0]);
   Mux(a=a[1],b=b[1],sel=sel,out=out[0]);
   Mux(a=a[2],b=b[2],sel=sel,out=out[0]);
   Mux(a=a[3],b=b[3],sel=sel,out=out[0]);
   Mux(a=a[4],b=b[4],sel=sel,out=out[0]);
   Mux(a=a[5],b=b[5],sel=sel,out=out[0]);
   Mux(a=a[6],b=b[6],sel=sel,out=out[0]);
   Mux(a=a[7],b=b[7],sel=sel,out=out[0]);
   Mux(a=a[8],b=b[8],sel=sel,out=out[0]);
   Mux(a=a[9],b=b[9],sel=sel,out=out[0]);
   Mux(a=a[10],b=b[10],sel=sel,out=out[0]);
   Mux(a=a[11],b=b[11],sel=sel,out=out[0]);
```

```
    Mux(a=a[12],b=b[12],sel=sel,out=out[0]);
    Mux(a=a[13],b=b[0],sel=sel,out=out[0]);
    Mux(a=a[14],b=b[0],sel=sel,out=out[0]);
}
```

1.2.7 OR 16

```
CHIP Or16 {
    IN a[16], b[16];
    OUT out[16];

    PARTS:
        Or(a=a[0],b=b[0],out=out[0]);
Or(a=a[1],b=b[1],out=out[1]);
Or(a=a[2],b=b[2],out=out[2]);
Or(a=a[3],b=b[3],out=out[3]);
Or(a=a[4],b=b[4],out=out[4]);
Or(a=a[5],b=b[5],out=out[5]);
Or(a=a[6],b=b[6],out=out[6]);
Or(a=a[7],b=b[7],out=out[7]);
Or(a=a[8],b=b[8],out=out[8]);
Or(a=a[9],b=b[9],out=out[9]);
Or(a=a[10],b=b[10],out=out[10]);
Or(a=a[11],b=b[11],out=out[11]);
Or(a=a[12],b=b[12],out=out[12]);
Or(a=a[13],b=b[13],out=out[13]);
Or(a=a[14],b=b[14],out=out[14]);
Or(a=a[15],b=b[15],out=out[15]);
}
```

1.2.8 NOT GATE

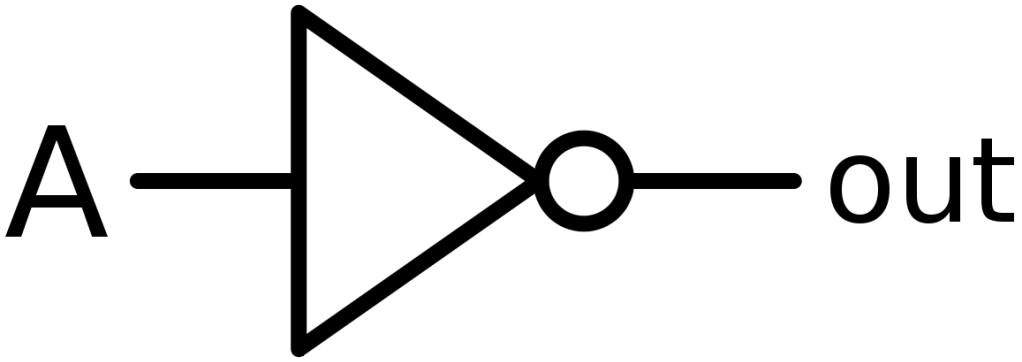


Figure 3 Simple Not Gate

CHIP Not {

IN in;

OUT out;

PARTS:

// Put your code here:

Nand(a=in,b=in,out=out);

}

1.2.9 AND GATE

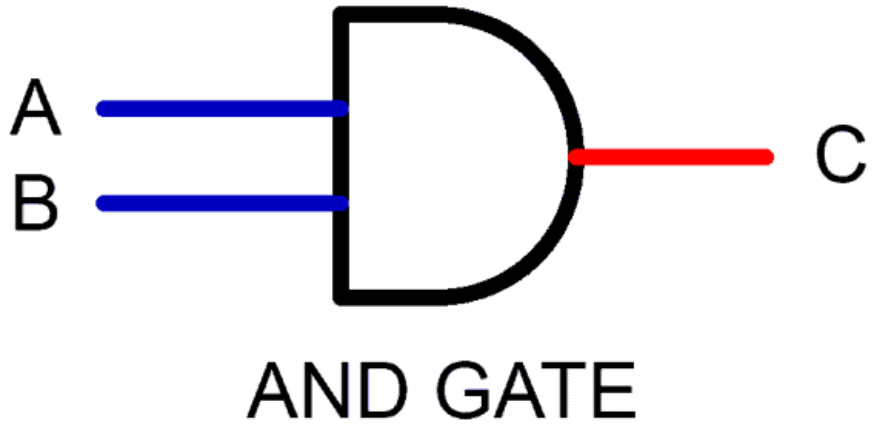


Figure 4 Simple And Gate

CHIP And {

IN a, b;

OUT out;

PARTS:

// Put your code here:

Nand(a=a,b=b,out=c);

Nand(a=c,b=c,out=out);

}

1.2.10 OR GATE

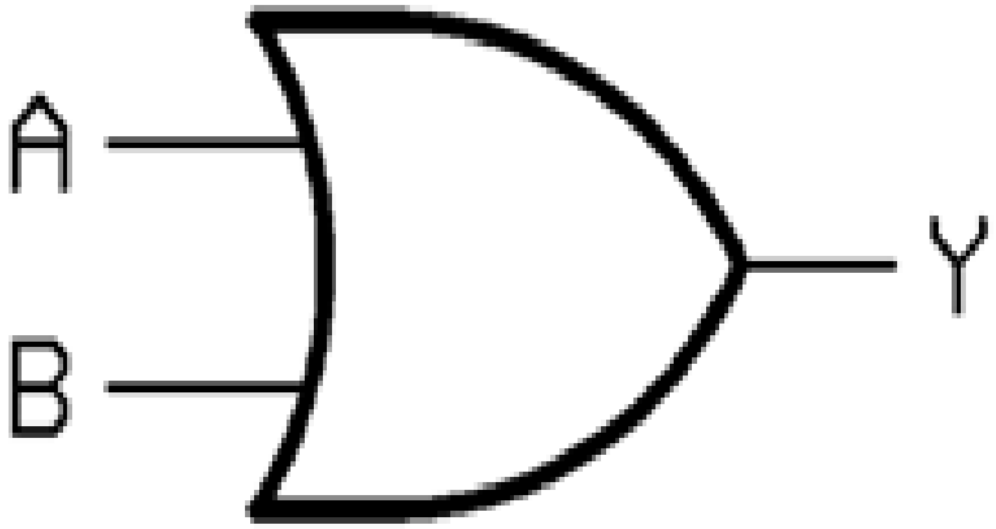


Figure 5 Simple Or Gate

CHIP Or {

IN a, b;

OUT out;

PARTS:

// Put your code here:

Nand(a=a,b=a,out=c1);

Nand(a=b,b=b,out=c2);

Nand(a=c1,b=c2,out=out);

}

1.2.11 MEMORY

The memory of the Hack Computer is entirely contained on the Memory chip. 16 bits are used. For general-purpose programme data storage, only Addresses 0 (0x000) through 16383 (0x3FFF) are utilised.

The built-in screen I/O subsystem is intended to use the registers at locations 16384 (0x4000) through 24575 (0x5FFF). Data written to addresses in this region result in output on the simulated 256 x 512 screen of the machine. The output from a keyboard connected to the computer running the Hack emulation programme controls the value of a single one word register at the last address in the RAM address space, 24576 (0x6000). The read-only nature of this keyboard memory map register.

The addresses 24577 (0x6001) through 32767 (0x7FFF) in the data memory are invalid.

DMux4Way, RAM16K, Screen, Keyboard, and Mux4Way16 are used to construct the memory.

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

    PARTS:
    // Put your code here:
     DMux4Way(in=load, sel=address[13..14], a=loadram1, b=loadram2, c=loadscreen, d=loadkbd);
    Or(a=loadram1, b=loadram2, out=loadram);
    RAM16K(in=in, load=loadram, address=address[0..13], out=ramout);
    Screen(in=in, load=loadscreen, address=address[0..12], out=scrout);
    Keyboard(out=kbout);
    Mux4Way16(a=ramout, b=ramout, c=scrout, d=kbout, sel=address[13..14], out=out);
}
```

1.2.12 RAM16K

```
CHIP RAM16K {
    IN in[16], load, address[14];
    OUT out[16];

    PARTS:
    // Put your code here:
```

DMux4Way(in=load,sel=address[12..13],a=a,b=b,c=c,d=d);

RAM4K(in=in,load=a,address=address[0..11],out=o1);

RAM4K(in=in,load=b,address=address[0..11],out=o2);

RAM4K(in=in,load=c,address=address[0..11],out=o3);

RAM4K(in=in,load=d,address=address[0..11],out=o4);

Mux4Way16(a=o1,b=o2,c=o3,d=o4,sel=address[12..13],out=out);

}

SCREEN

256 × 512 black-and-white pixels are continuously refreshed by the integrated screen memory map chip. 32 consecutive 16-bit words make up each row.

KEYBOARD

The code of the presently pressed key is output by the built-in keyboard memory map.

1.2.13 DMUX4WAY

A 4 way demultiplexor with 4 outputs is the DMux4Way.

CHIP DMux4Way {

IN in, sel[2];

OUT a, b, c, d;

PARTS:

// Put your code here:

DMux(in=in,sel=sel[1],a=a0,b=b0);

DMux(in=a0,sel=sel[0],a=a,b=b);

DMux(in=b0,sel=sel[0],a=c,b=d);

}

1.2.14 DMUX4WAY16

A 4-way 16 bit multiplexor is the Mux4Way16.

```
CHIP Mux4Way16 {
    IN a[16], b[16], c[16], d[16], sel[2];
    OUT out[16];

    PARTS:
    // Put your code here:
Mux16(a=a,b=b,sel=sel[0],out=o1);
Mux16(a=c,b=d,sel=sel[0],out=o2);
Mux16(a=o1,b=o2,sel=sel[1],out=out);
}
```

1.3 EXPERIMENT AND RESULTS

Instruction Memory (ROM32K), the Central Processing Unit (CPU), and Data Memory (RAM) are the three essential components of the Hack Computer. The Hack Computer only stores data in 16-bit words. As was previously said, the CPU, RAM, and Instruction Memory are all built-in.

A built-in clock regulates the computer's progressive operation. The instruction at the ROM location of the current value of the PC is decoded at the start of a clock cycle. The ALU executes the requested computation, and the outcome is saved. The programme kept in the computer's ROM is run when reset is 0. When reset is set to 1, the program's execution is restarted, and PC value is set to 0. The present programme will continue to run when reset is zero once more.

1.3.1 HDL CODE

```
CHIP Computer {

    IN reset;

    PARTS:
    // Put your code here:
     ROM32K(address=pc, out=instruction);
     CPU(inM=memOut, instruction=instruction, reset=reset, outM=outM,
```

```
    writeM=writeM, addressM=addressM, pc=pc);
  Memory(in=outM, load=writeM, address=addressM, out=memOut);
}
```

1.3.2 TST FILE

The computer-generated tst file.Hdl is a programme that will insert the numbers 1, 2, and 3 at position 30 and then calculate their sum. A.hack file is used to store the machine code translation of the Assembly code. This file is loaded into the ROM32K, which houses the instructions. A predetermined number of clock cycles are read out from the procedure. To restart, we set the reset value to 1.

```
load Computer.hdl,
output-file Computermod.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 RAM16K[2]%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 mod.hack,
output;
set RAM16K[0] 9,
set RAM16K[1] 2,
//output;

repeat 50 {
   tick, tock, output;
}

set reset 1,
set RAM16K[0] 0,
set RAM16K[1] 0,
tick,tock,output;
```

1.4 OUTPUT

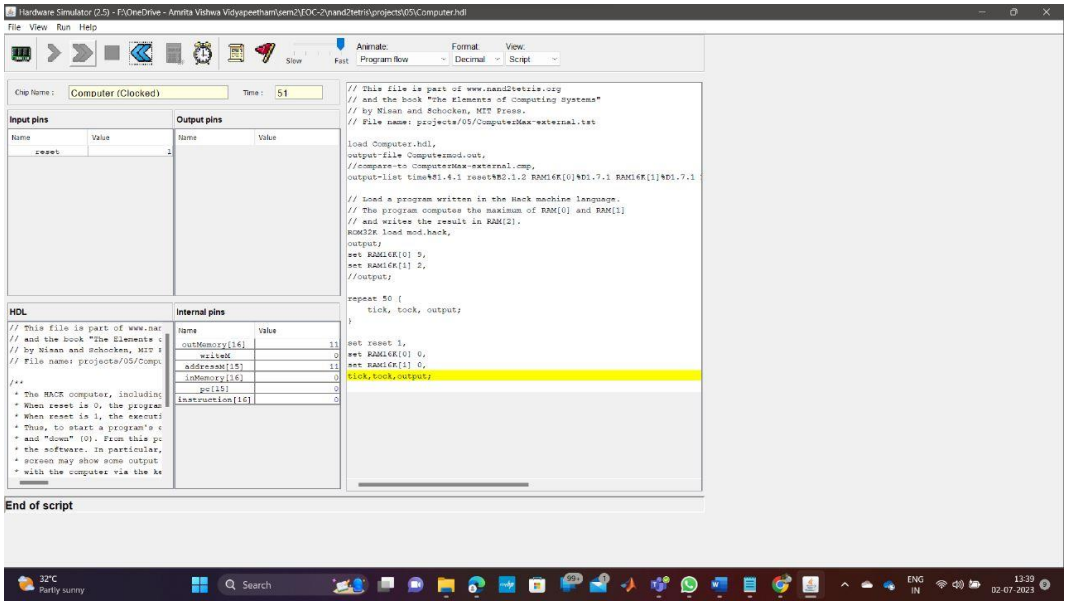


Figure 6 Part A output-1

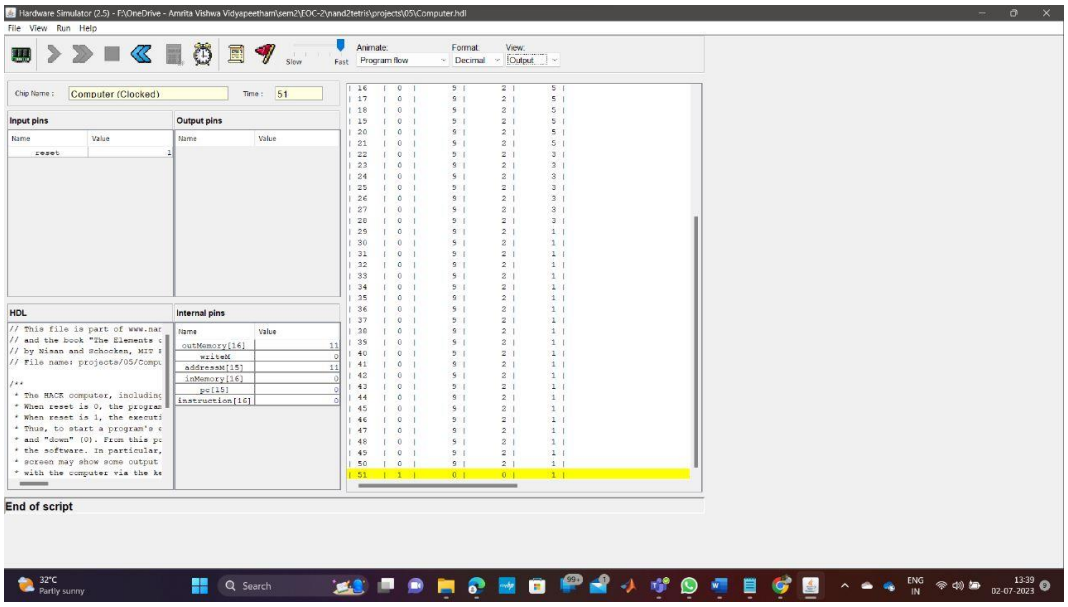


Figure 7 Part A output-2

2. PART B

2.1 INTRODUCTION

Our project focuses on the implementation of a single-player Pong game using the Jack programming platform. Pong is a classic arcade game where players control a paddle to hit a ball back and forth, trying to prevent the ball from passing their paddle. Our version of Pong incorporates several features and allows the user to customize the game by selecting the difficulty level, ball design, and bat design.

The project consists of four main code files: Ball, Player, Game, and Main. The Ball file offers the user the option to choose from three pre-existing ball designs. It also includes the code that defines the movement of the ball within the game. The Player file enables the user to select the design of the paddle (bat) and determines the movement of the paddle during gameplay. The Game file serves as the main component that governs the functioning of the game. It is based on the working methodology of Pong, providing the necessary game logic and controls to ensure an engaging gaming experience. The Main file acts as the entry point for the game, calling the other three files and enabling the VMemulator to run the code seamlessly.

When running the code, the user is prompted to select the difficulty level from options such as easy, difficult, and insane. Once the difficulty level is chosen, the user can further customize their gaming experience by selecting the ball design and paddle design from the pre-programmed options. Once the game begins, the user controls the paddle's movement using the left and right arrow keys. In this single-player version of Pong, the objective is not to win against an opponent but rather to increase the score each time the ball touches the paddle. The ball bounces off the paddle and the walls, which enclose three sides of the screen (top, left, and right). However, if the paddle misses the ball and it touches nothing, the game is over.

In this project report, we will discuss in detail the implementation of the Pong game, including the functionalities of each code file, the game flow, and the user interactions. We will also address the challenges encountered during the development process and the strategies used to overcome them. Additionally, we will provide insights into the lessons learned, potential improvements, and our overall experience in implementing the single-player Pong game using the Jack programming platform.

2.2 METHODOLOGY

2.2.1 Ball

Code:

```
class Ball{
   field int x, y, radius, ballDesign;
   constructor Ball new(int _x,int _y, int r, int design){
      let x = _x;
      let y = _y;
      let radius = r;
      let ballDesign = design;
      return this;
   }
   method int getX(){return x;}
   method int getY(){return y;}
   method int getR(){return radius;}
   method void move(int dx, int dy){ // to present
      do hide();
      let x = x + dx;
      let y = y + dy;
      do draw();
      return;
   }
   method void hide(){
      do Screen.setColor(false);
      do Screen.drawCircle(x, y, radius);
      return;
   }
   method void draw(){ // to present
      if (ballDesign = 1){ // Standard design
         do Screen.setColor(true);
         do Screen.drawCircle(x, y, radius);
      }else{
         if(ballDesign = 2){ // Concentric circles
```

```
            do Screen.setColor(true);
            do Screen.drawCircle(x, y, radius);
            do Screen.setColor(false);
            do Screen.drawCircle(x, y, radius - 3);
            do Screen.setColor(true);
            do Screen.drawCircle(x, y, radius - (radius - 1));
         }else{
            if(ballDesign = 3){ // Rectangle in circle
               do Screen.setColor(true);
               do Screen.drawCircle(x, y, radius);
               do Screen.setColor(false);
               do Screen.drawRectangle(x - (radius/3+2), y - (radius/3+2),x +
(radius/3+2),y + (radius/3+2));
            }else{
               do Screen.setColor(true);
               do Screen.drawCircle(x, y, radius);
            }
         }
      }

      return;
   }
   method void dispose() {
      do Memory.deAlloc(this);
      return;
   }
}
```

Code Explanation:

1.The Ball class represents a ball object in the Pong game. It has four fields: x and y (coordinates of the ball), radius (the radius of the ball), and ballDesign (an integer representing the design type of the ball).

2.The constructor Ball new(int _x, int _y, int r, int design) is responsible for creating a new instance of the Ball object. It takes four parameters: _x and _y (the initial coordinates of the ball), r (the radius), and design (the design type). It initializes the object's fields and returns the newly created object.

3.The getX(), getY(), and getR() methods are getter methods that return the values of the x, y, and radius fields, respectively.

4.The move(int dx, int dy) method is used to move the ball by updating its coordinates. It takes two parameters dx and dy (representing the change in the x and y coordinates) and moves the ball accordingly. Before updating the position, it hides the ball by calling the hide() method, then updates the coordinates, and finally calls the draw() method to redraw the ball in the new position.

5.The hide() method sets the drawing color to false (representing an erased state) and draws a circle with the current ball's coordinates and radius using the Screen.drawCircle() function.

6.The draw() method is responsible for drawing the ball on the screen based on its design type. It uses conditional statements (if and else) to check the value of ballDesign and draw the ball accordingly. For each design type, it sets the drawing color to true (representing an active state) and uses the Screen.drawCircle() and Screen.drawRectangle() functions to draw circles and rectangles on the screen.

7.The dispose() method deallocates the memory occupied by the Ball object using the Memory.deAlloc() function.

8.Overall, this code provides a basic implementation of the Ball class in Jack, allowing you to create and manipulate ball objects in the Pong game.

2.2.2 Game

Code:

```
class Game{
   field int playerDx, compDx, ballDx, ballDy, pause, score, gameMode, ballDesign, playerDesign;
   field Player me, comp;
   field Ball ball;
   field boolean gameOver;
   constructor Game new(){
      let ballDx = 3;
      let ballDy = -4;
      let compDx = 3;
      let playerDx = 3;
      let score = 0;
      return this;
   }
   method void printScore(){
```

```
    do Output.moveCursor(22,7);
        do Output.printInt(score);
    return;
  }
  method void wallCollision(Ball ball){ // to present
    if((ball.getX()+ball.getR()) > (511 - ball.getR() - Math.abs(ballDx))){
      let ballDx = -ballDx;
      do ball.move(ballDx ,ballDy);
    }else{}
    if((ball.getX()-ball.getR()) < (0 + ball.getR() + Math.abs(ballDx))){
      let ballDx = -ballDx;
      do ball.move(ballDx, ballDy);
    }else{}
    return;
  }
  method void playerCollision(Ball ball, Player player, Player computer){ // to
present
    if((ball.getX() > computer.getX()) & (ball.getX() < (computer.getX() +
computer.getLength()))){
      if((ball.getY() - ball.getR() - Math.abs(ballDy) - 1) < (computer.getY() +
computer.getHeight())){
        let ballDy = -ballDy;
        do ball.move(ballDx,ballDy);
      }else{}
    }else{}
    if((ball.getX() > player.getX()) & (ball.getX() < (player.getX() +
player.getLength()))){
      if((ball.getY() + ball.getR() + Math.abs(ballDy) + 1) > player.getY()){
        let ballDy = -ballDy;
        let score = score + 1;
        do ball.move(ballDx, ballDy);
      }else{}
    }else{}
    return;
```

```
}
method boolean over(Ball ball){
   if(((ball.getY() + ball.getR()) > 235) | ((ball.getY() - ball.getR()) < 15)){
      return true;
   }else{return false;}
}
method void printGameModes(){
   do Output.moveCursor(2,16);
         do Output.printString("Select game mode, (press # and ENTER)");
   do Output.moveCursor(6,11);
         do Output.printString("1");
   do Output.moveCursor(8,10);
         do Output.printString("EASY");
   do Output.moveCursor(6,30);
         do Output.printString("2");
   do Output.moveCursor(8,29);
         do Output.printString("HARD");
   do Output.moveCursor(6,49);
         do Output.printString("3");
   do Output.moveCursor(8,46);
         do Output.printString("INSANE");
   return;
}
method void printBallDesigns(){
   do Output.moveCursor(2,16);
         do Output.printString("Select ball design, (press # and ENTER)");
   do Output.moveCursor(6,11);
         do Output.printString("1");
   do Screen.setColor(true);
   do Screen.drawCircle(90,100,10);

   do Output.moveCursor(6,30);
```

```
        do Output.printString(" 2");
    do Screen.setColor(true);
    do Screen.drawCircle(250, 100, 10);
    do Screen.setColor(false);
    do Screen.drawCircle(250, 100, 5);
    do Screen.setColor(true);
    do Screen.drawCircle(250, 100, 1);

    do Output.moveCursor(6,49);

        do Output.printString(" 3");
    do Screen.setColor(true);
    do Screen.drawCircle(400, 100, 10);
    do Screen.setColor(false);
    do Screen.drawRectangle(400-4, 100-4,400+4,100+4);
    return;
}
method void printPlayerDesigns(){
    do Output.moveCursor(2,16);
        do Output.printString("Select player design, (press # and ENTER)");
    do Output.moveCursor(6,11);
        do Output.printString("1");
    do Screen.setColor(true);
    do Screen.drawRectangle(50, 100, 130, 100 + 8);

    do Output.moveCursor(6,30);
        do Output.printString("2");
    do Screen.setColor(true);
    do Screen.drawRectangle(210, 100, 290, 100 + 1);
    do Screen.drawRectangle(210, 100 + 7, 290, 100 + 8);
    do Screen.drawLine(210, 100+4,290,100+4);

    do Output.moveCursor(6,49);
```

```
            do Output.printString("3");
        do Screen.drawRectangle(360, 100, 370, 100+8);
        do Screen.drawRectangle(430, 100, 440, 100+8);
        do Screen.drawRectangle(360, 100, 440, 100+2);
        return;
    }
    method void start(){ // to present
        var int keyPressed, _keyPressed;
        var boolean pressed;
        let pressed = false;
        let keyPressed = 0;
        do printGameModes();

        while(~pressed){
            do Output.moveCursor(13,21);
            let _keyPressed = Keyboard.readInt("Enter game mode: ");
            if(~(_keyPressed = 0)){
                let keyPressed = _keyPressed;
                let pressed = true;
            }
        }
        do Screen.clearScreen();
        let gameMode = keyPressed;
        if(~(gameMode = 3)){
            let pressed = false;
        }
        do printBallDesigns();

        while(~pressed){
            do Output.moveCursor(13,21);
            let _keyPressed = Keyboard.readInt("Enter ball design: ");
            if(~(_keyPressed = 0)){
                let keyPressed = _keyPressed;
```

```
            let pressed = true;
        }
    }
    do Screen.clearScreen();
    let ballDesign = keyPressed;
    do printPlayerDesigns();
    let pressed = false;

    while(~pressed){
        do Output.moveCursor(13,21);
        let _keyPressed = Keyboard.readInt("Enter player design: ");
        if(~(_keyPressed = 0)){
            let keyPressed = _keyPressed;
            let pressed = true;
        }
    }
    do Screen.clearScreen();
    let playerDesign = keyPressed;
    return;
}
method void setMode(){
    if(gameMode = 1){
        let me = Player.new(220,220, 50, playerDesign);
        let comp = Player.new(0,10, 511, 1);
        let ball = Ball.new(150,150, 8, ballDesign);
        let pause = 50;
    }else{
        if(gameMode = 2){
            let me = Player.new(220,220, 40, playerDesign);
            let comp = Player.new(0,10, 511, 1);
            let ball = Ball.new(150,150, 5, ballDesign);
            let pause = 20;
        }else{
```

```
            if(gameMode = 3){
               let me = Player.new(220,220, 25, playerDesign);
               let comp = Player.new(0,10, 511, 1);
               let ball = Ball.new(150,150, 3, 1);
               let pause = 15;
            }else{
               let me = Player.new(220,220, 50, playerDesign);
               let comp = Player.new(0,10, 511, 1);
               let ball = Ball.new(150,150, 8, ballDesign);
               let pause = 50;
            }
         }
      }
      return;
   }
   method void play(){ // to present
      var char keyPressed, _keyPressed;
      let keyPressed = 0;
      do ball.draw();
      do comp.draw();
      do me.draw();
      do Output.moveCursor(22,0);
            do Output.printString("SCORE: ");
      while(~gameOver){
         do playerCollision(ball, me, comp);
         do wallCollision(ball);
         do printScore();
         do ball.move(ballDx, ballDy);
         let _keyPressed = Keyboard.keyPressed();
         if(~(_keyPressed = 0)){
            let keyPressed = _keyPressed;
         }
         if(keyPressed = 130){
```

```
            do me.move(-playerDx);
        }else{}
        if(keyPressed = 132){
            do me.move(playerDx);
        }else{}
        if(over(ball)){
            do Output.moveCursor(11,25);
                do Output.printString("Game Over!");
            do Sys.wait(3000 - pause);
            let gameOver = true;
        }
        do Sys.wait(pause);
      }
      do me.dispose();
      do comp.dispose();
      do ball.dispose();
      return;
   }
   method void dispose() {
      do Memory.deAlloc(this);
      return;
   }
}
```

Code Explanation:

1.This code snippet appears to be a simplified implementation of a game using a programming language that includes classes and object-oriented concepts. Here's a breakdown of the code:

2.The Game class represents the game itself and contains various fields and methods for managing the game state.

3.The class has several fields such as playerDx, compDx, ballDx, ballDy, pause, score, gameMode, ballDesign, playerDesign, me, comp, ball, and gameOver. These fields store different properties and objects related to the game.

4.The constructor new() initializes some of the fields with default values and returns an instance of the Game class.

5.The printScore() method prints the current score on the screen.

6.The wallCollision() method handles the collision of the ball with the walls of the game area and updates its position accordingly.

7.The playerCollision() method handles the collision of the ball with the players (controlled by the user and the computer) and updates the ball's position and score accordingly.

8.The over() method checks if the ball has reached the top or bottom boundary, indicating that the game is over.

9.The printGameModes(), printBallDesigns(), and printPlayerDesigns() methods are responsible for displaying the available game modes, ball designs, and player designs, respectively.

10.The start() method allows the player to select the game mode, ball design, and player design before the game begins.

11.The setMode() method sets up the game mode based on the selected options.

12.The play() method represents the main gameplay loop, where the ball moves, collisions are detected, and user input is handled. The game continues until it's over.

13.The dispose() method is used to deallocate memory or resources associated with the game.

14.Overall, this code provides a basic structure for a game with players, a ball, and different game modes and designs. However, some parts of the code, such as variable assignments and conditions, may not function as intended due to incorrect usage of variable scoping. Additionally, some parts of the code are incomplete and lack implementation details.

2.2.3 Player

Code:

```
class Player{
   field int x, y;
   field int length, height, design;
   constructor Player new(int _x, int _y, int _length, int playerDesign){
      let x = _x;
      let y = _y;
      let length = _length;
      let height = 8;
      let design = playerDesign;
      return this;
   }
   method int getY(){return y;}
```

```
   method int getX(){return x;}
   method int getLength(){return length;}
   method int getHeight(){return height;}
   method void move(int dx){ // to present
      if((x + dx) < 0){
         return;
      }else{
         if(((x+length) + dx) > 511){
            return;
         }else{}
      }
      if(design = 1){
         if(dx > 0){
            do Screen.setColor(false);
            do Screen.drawRectangle(x, y, (x + dx), (y + height));
            do Screen.setColor(true);
            do Screen.drawRectangle((x + length), y, (x + length + dx), (y +
height));
         }else{
            if(dx < 0){
               do Screen.setColor(false);
               do Screen.drawRectangle((x + length + dx), y, (x + length), (y +
height));
               do Screen.setColor(true);
               do Screen.drawRectangle((x+dx), y, x, (y + height));
            }else{}
         }
      }else{
         if(design = 2){
            if(dx > 0){
               do Screen.setColor(false);
               do Screen.drawRectangle(x, y, (x + dx), y + 1);
               do Screen.drawRectangle(x, (y + height -1), (x + dx), (y + height));
               do Screen.drawLine(x,(y+ 4),(x + dx), (y+ 4));
```

```
            do Screen.setColor(true);
            do Screen.drawRectangle((x + length), y, (x + length + dx), (y+1));
            do Screen.drawRectangle((x + length), (y + height - 1), (x + length +
dx), (y + height));
            do Screen.drawLine((x+length), (y+ 4), (x+length + dx), (y+4));
         }else{
            if(dx < 0){
               do Screen.setColor(false);
               do Screen.drawRectangle((x+length + dx), y, (x+length), (y+1));
               do Screen.drawRectangle((x+length + dx), (y+height - 1), (x +
length), (y + height));
               do Screen.drawLine((x + length), (y + 4), (x + length + dx), (y+4));
               do Screen.setColor(true);
               do Screen.drawRectangle((x + dx), y, x, y+1);
               do Screen.drawRectangle((x + dx), (y + height - 1), x, (y +
height));
               do Screen.drawLine((x+ dx), (y+4), x, (y+4));
            }else{}
         }
      }else{
         if(design = 3){
            if(dx > 0){
               do Screen.setColor(false);
               do Screen.drawRectangle(x,y, (x + dx), (y + height));
               do Screen.drawRectangle((x + length - 10), (y + 3), (x + length -
10 + dx), (y + height));
               do Screen.setColor(true);
               do Screen.drawRectangle((x + 10), y, (x + 10 + dx), (y + height));
               do Screen.drawRectangle((x + length), y, (x + length + dx), (y +
height));

            }else{
               if(dx < 0){
                  do Screen.setColor(false);
                  do Screen.drawRectangle((x + length + dx), y, (x + length), (y +
```

```
height));
                    do Screen.drawRectangle((x + 10 + dx), (y + 3), (x + 10), (y +
height));
                    do Screen.setColor(true);
                    do Screen.drawRectangle((x + dx), y, x, (y + height));
                    do Screen.drawRectangle((x + length - 10 + dx), y, (x + length -
10), (y + height));

                }else{}
              }
            }else{
              if(design = 4){
                if(dx > 0){
                    do Screen.setColor(false);
                    do Screen.drawRectangle(x,y, (x + dx), (y + height));
                    do Screen.drawRectangle((x + length - 10), y, (x + length - 10 +
dx), (y + 5));
                    do Screen.setColor(true);
                    do Screen.drawRectangle((x + 10), y, (x + 10 + dx), (y +
height));
                    do Screen.drawRectangle((x + length), y, (x + length + dx), (y +
height));

                }else{
                    if(dx < 0){
                        do Screen.setColor(false);
                        do Screen.drawRectangle((x + length + dx), y, (x + length), (y
+ height));
                        do Screen.drawRectangle((x + 10 + dx), y, (x + 10), (y + 5));
                        do Screen.setColor(true);
                        do Screen.drawRectangle((x + dx), y, x, (y + height));
                        do Screen.drawRectangle((x + length - 10 + dx), y, (x +
length - 10), (y + height));

                    }else{}
```

```
                }
            }
        }
    }
}

    let x = x + dx;
    return;
}
method void draw(){ // to present
    do Screen.setColor(false);
    if(design = 1){
        do Screen.drawRectangle(x, y, (x+length), (y+height));
    }else{
        if(design = 2){
            do Screen.drawRectangle(x,y,(x+length), (y + 1));
            do Screen.drawRectangle(x, (y + height - 1), (x+length), (y+height));
            do Screen.drawLine(x, (y + 4), (x + length), (y+4));
        }else{
            if(design = 3){
                do Screen.drawRectangle(x, y, (x + 10), (y + height));
                do Screen.drawRectangle((x + length - 10), y, (x + length), (y +
height));
                do Screen.drawRectangle(x,y,(x + length), (y + 2));
            }else{
                do Screen.drawRectangle(x, y, (x + 10), (y + height));
                do Screen.drawRectangle((x + length - 10), y, (x + length), (y +
height));
                do Screen.drawRectangle(x,(y + 6),(x + length), (y + height));
            }
        }
    }

    return;
```

```
    }
    method void dispose() {
        do Memory.deAlloc(this);
        return;
    }
}
```

Code Explanation:

1.The Player class has several fields that store information about the player's position (x and y), dimensions (length and height), and design.

2.The class has a constructor called new which is used to create a new instance of the Player class. It takes four parameters: _x, _y, _length, and playerDesign. Inside the constructor, the values of the parameters are assigned to the corresponding fields of the Player object. The height field is set to 8, and the design field is set to the value of the playerDesign parameter. Finally, the constructor returns the created Player object.

3.The class has several getter methods (getY, getX, getLength, getHeight) that allow access to the private fields of the Player object.

4.The move method takes an integer parameter dx, which represents the amount of movement in the x-axis. The purpose of this method seems to be to update the player's position based on the provided dx value. The method contains a series of conditions that check if the new position (x + dx) is within certain bounds (0 to 511). If the new position is within bounds, the method proceeds to update the player's position and draw the appropriate design on the screen.

5.The draw method is responsible for drawing the player on the screen based on its design. It uses the Screen class (presumably defined elsewhere) to draw rectangles and lines to represent the player's appearance. The specific design is determined by the value of the design field.

6.The dispose method seems to be used for memory management or resource cleanup. It calls Memory.deAlloc (presumably defined elsewhere) to deallocate memory associated with the current Player object.

7.Overall, this code represents a basic player class that stores information about a player's position, dimensions, and design. It provides methods to move the player, draw it on the screen, and dispose of it when no longer needed. However, the code snippet seems incomplete and lacks some context about how it fits into the larger program or game it belongs to.

2.2.4 Main

Code:

```
class Main{
    function void main(){
```

```
        var Game game;
        do Screen.clearScreen();

        let game = Game.new();
        do game.start();
        do game.setMode();
        do game.play();

        do game.dispose();
        return;
    }
}
```

Code Explanation:

1.The provided code represents a Main class with a main function. In many programming languages, the main function serves as the entry point of the program, meaning that it is the first function to be executed when the program starts. Let's go through the code and explain its different parts:

2.The Main class has a main function. Inside the main function, a variable called game of type Game is declared. The Game class is not defined in the provided code, so it is assumed that it exists elsewhere. The Screen.clearScreen() function is called, which presumably clears the screen or sets up the graphical environment for the game.

3.The new() function is called on the Game class to create a new instance of the Game object. The let keyword is used to assign the created object to the game variable.

4.The start(), setMode(), and play() methods are called on the game object. These methods likely belong to the Game class and are responsible for initializing the game, setting its mode, and starting the gameplay, respectively.

5.The dispose() method is called on the game object. This method likely handles any necessary cleanup or resource deallocation related to the Game object.

6.The main function ends with a return statement, indicating the end of the function and subsequently the end of the program execution.

7.Overall, this code snippet represents the entry point of the program and demonstrates a basic structure for starting and running a game. It creates a Game object, initializes it, sets the game mode, and starts the gameplay. Finally, it disposes of the Game object before exiting the program. However, without the implementation of the Game class and its associated methods, it is difficult to provide a more detailed explanation of the overall functionality of the program.

2.3 EXPERIMENTS & RESULTS:

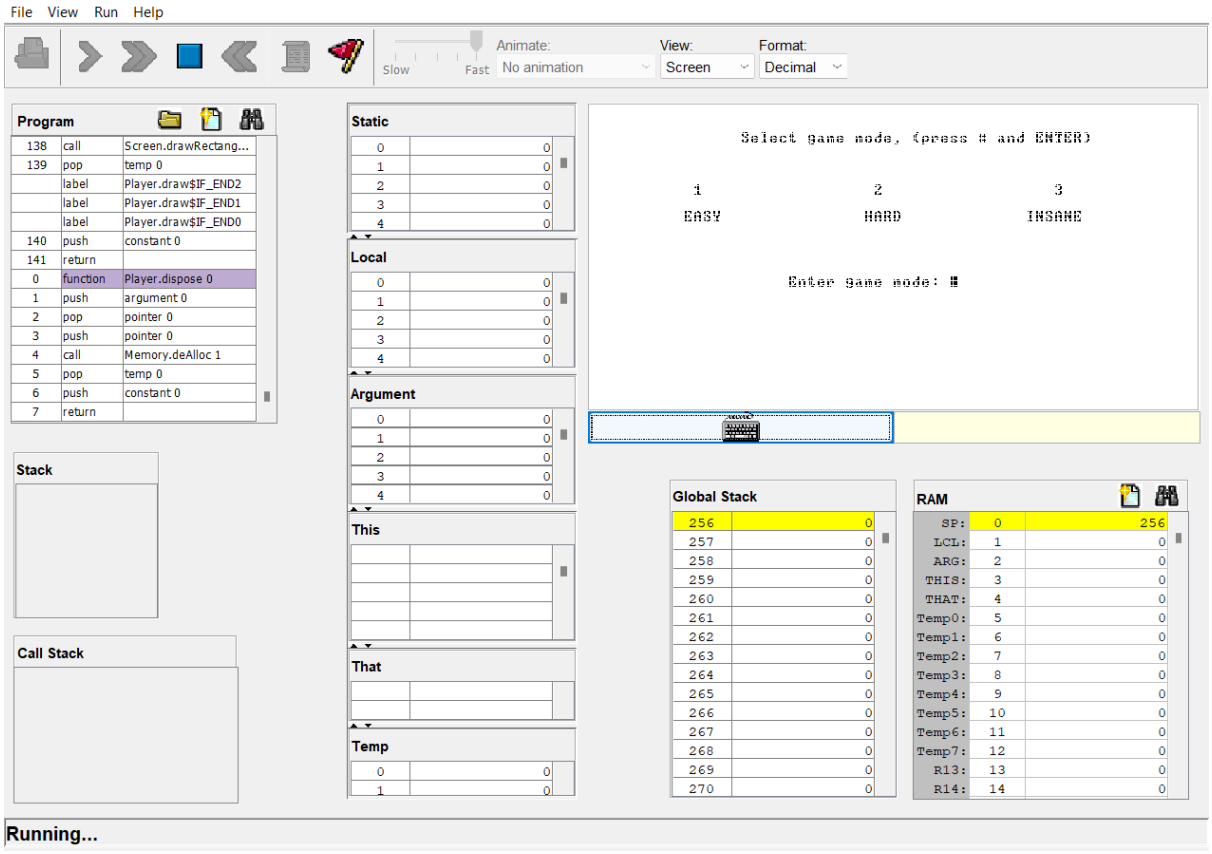


Figure 8 Part B output-1

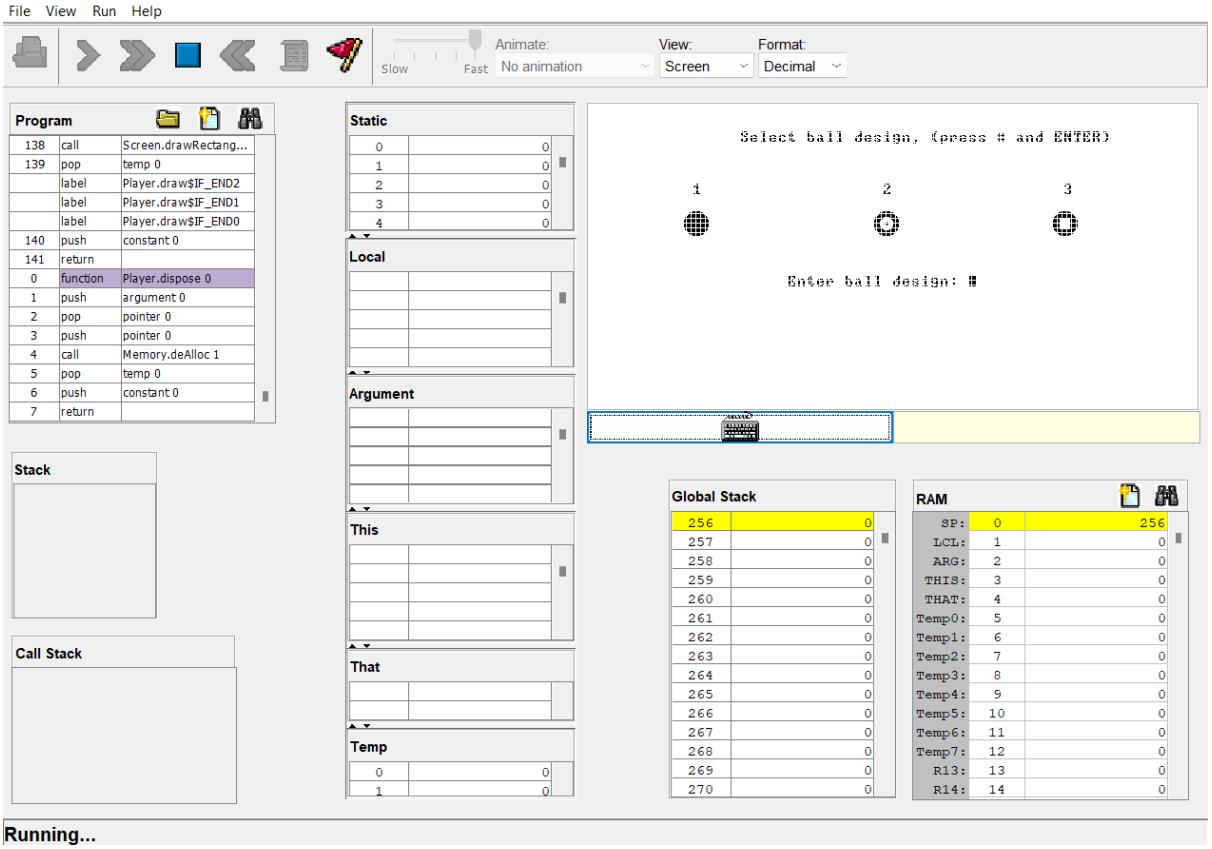


Figure 9 Part B output-2

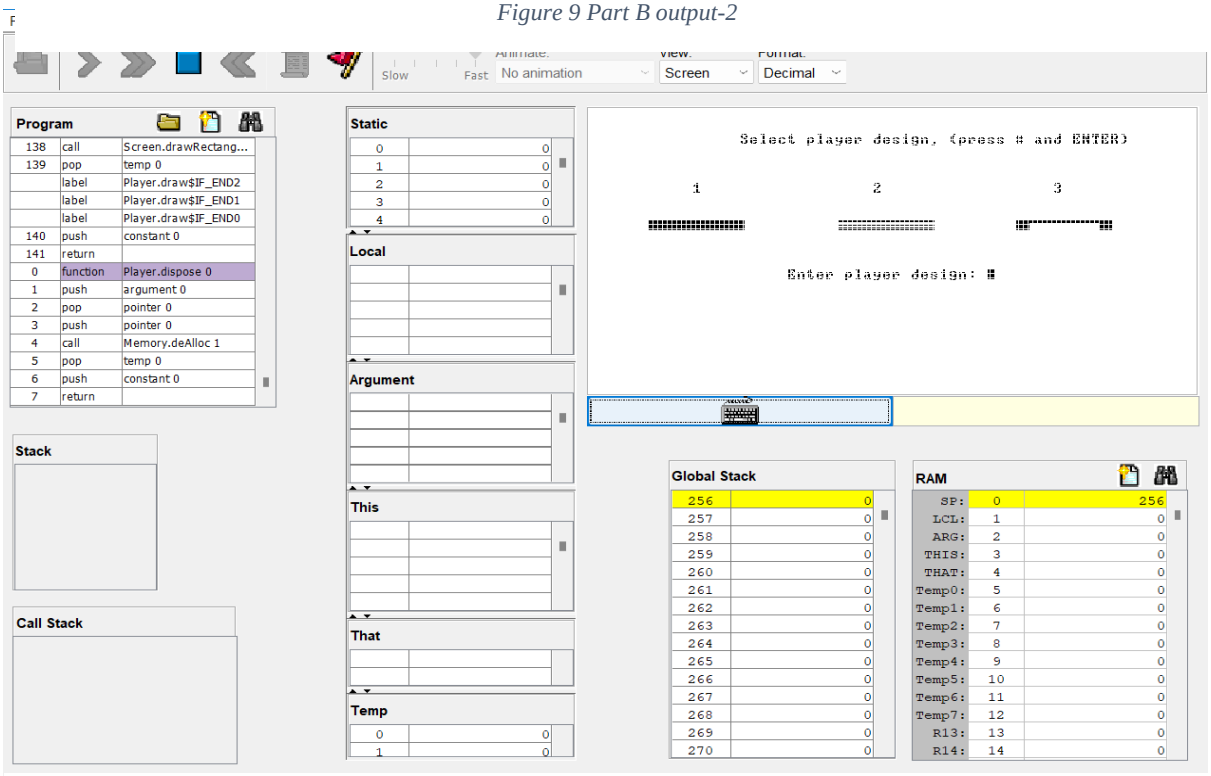


Figure 10 Part B output-3

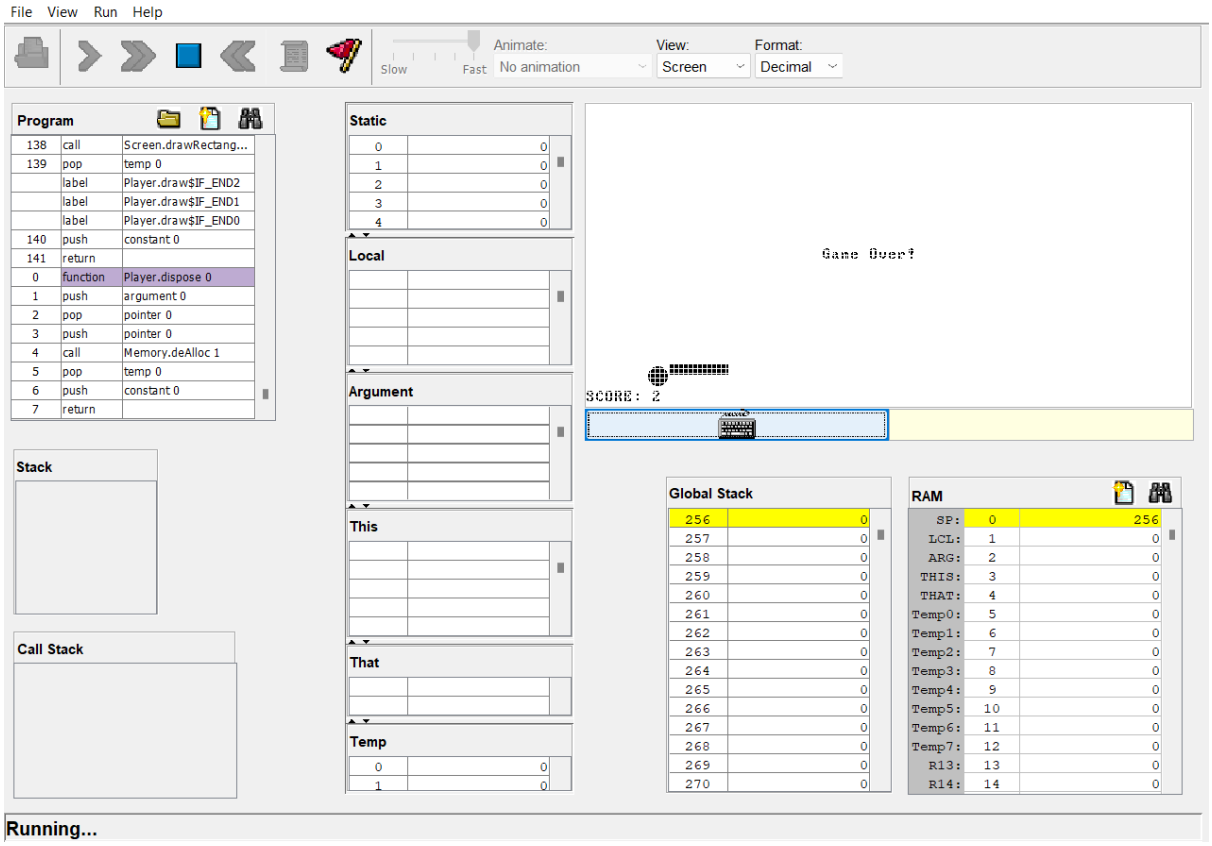


Figure 10 Part B output-4

3. PART C

3.1INTRODUCTION

Our project focused on the implementation of a single-player Hand Cricket game against a computer opponent using the Jack programming platform. Hand Cricket is a popular game where players simulate a cricket match using their hands as a bat and attempting to score runs against their opponent's bowling. In our version, the player competes against the computer, which throws random numbers as its bowling actions. The objective of the game is to outscore the computer and achieve the highest score possible.

The project consists of three main code files: Clear Screen, RandomNumberGenerator, and Main. The Clear Screen file contains a function specifically designed to clear the screen by printing a series of empty spaces, ensuring a clean and visually appealing interface. The RandomNumberGenerator file provides various functions, with the 'rand' function being the most essential. This function generates a random integer between two specified arguments,

simulating the computer's bowling actions during the game. The Main file contains the core code of the game, incorporating the working methodology of Hand Cricket against a computer opponent.

To determine the batting order, the game starts with a toss. Instead of the traditional heads or tails, we have introduced a toss based on the concept of odd or even. The player and the computer make their respective choices, and the winner of the toss automatically gets the opportunity to bat first. The game then progresses similarly to a regular Hand Cricket match, with the player attempting to score runs by avoiding the computer's random numbers while batting. The objective is to outperform the computer's score and emerge as the victor.

In this project report, we will discuss the implementation details of the game, including the functionalities of each code file, the game flow, and the user interactions. We will also highlight the challenges we faced during the development process and how we overcame them. Additionally, we will provide insights into the lessons learned, improvements that can be made, and our overall experience in implementing the single-player Hand Cricket game using the Jack programming platform.

3.2METHODOLOGY

3.2.1ClearScreen

Code:

```
class ClearScreen {
   function void clearScreen() {
      var int i;
      let i = 0;
      while (i < 24) {
         do Output.printString("                                                ");
         let i = i + 1;
      }
      do Output.moveCursor(0, 0);
      return;
   }

   function void printEmptyLines(int numLines) {
      var int i;
      let i = 0;
```

```
        while (i < numLines) {
            do Output.println();
            let i = i + 1;
        }
        return;
    }
}
```

Code Explanation:

1.clearScreen() method:

This method is responsible for clearing the screen by printing empty lines.

It uses a variable i to keep track of the number of lines printed.

It initializes i to 0 and enters a while loop that continues until i is less than 24 (the number of lines to clear).

Within the loop, it uses Output.printString() to print a string consisting of 64 whitespace characters. This effectively clears the line.

After printing the empty line, it increments i by 1 using let i = i + 1;.

Finally, it calls Output.moveCursor(0, 0) to move the cursor to the top-left corner of the screen.

The method then returns.

2.printEmptyLines(int numLines) method:

This method is responsible for printing a specified number of empty lines.

It takes an integer parameter numLines which indicates the number of empty lines to print.

It uses a variable i to keep track of the number of lines printed.

It initializes i to 0 and enters a while loop that continues until i is less than numLines.

Within the loop, it uses Output.println() to print an empty line.

After printing the line, it increments i by 1 using let i = i + 1;.

The loop continues until the desired number of empty lines are printed.

The method then returns.

3.Overall, the ClearScreen class provides a way to clear the screen by printing empty lines and also allows printing a specific number of empty lines. This code can be used in a console-based application or simulation to create a clear and organized display.

3.2.2 RandomNumberGenerator

Code:

```
class RandomNumberGenerator {

   function int rand(int min, int max,int count) {
   if((min = 1)&(max = 6)){
      if(RandomNumberGenerator.remainder(count,6)=0){
      return 3;
      }
      if(RandomNumberGenerator.remainder(count,6)=1){
      return 2;
      }
      if(RandomNumberGenerator.remainder(count,6)=2){
      return 5;
      }
      if(RandomNumberGenerator.remainder(count,6)=3){
      return 1;
      }
      if(RandomNumberGenerator.remainder(count,6)=4){
      return 6;
      }
      if(RandomNumberGenerator.remainder(count,6)=5){
      return 4;
      }
   }
   else{
      if(RandomNumberGenerator.remainder(count,7)=0){
      return 3;
      }
      if(RandomNumberGenerator.remainder(count,7)=1){
      return 0;
      }
      if(RandomNumberGenerator.remainder(count,7)=2){
```

```
    return 2;
    }
    if(RandomNumberGenerator.remainder(count,7)=3){
    return 5;
    }
    if(RandomNumberGenerator.remainder(count,7)=4){
    return 1;
    }
    if(RandomNumberGenerator.remainder(count,7)=5){
    return 6;
    }
    if(RandomNumberGenerator.remainder(count,7)=6){
    return 4;
    }

}
return 3;
}

function boolean lt(int num1, int num2){
if(num1<num2){
return true;
}
else {return false;}
}

function boolean gt(int num1, int num2){
if(num1>num2){
return true;
}
else {return false;}
}
```

```
function boolean eq(int num1, int num2){
if(num1=num2){
return true;
}
else{return false;}
}

function boolean isEven(int num) {
while (RandomNumberGenerator.gt(num, 1)) {
let num = num - 2;
}

return (RandomNumberGenerator.eq(num, 0));
}



function int remainder(int x, int y) {
while (RandomNumberGenerator.gt(x, y)) {
let x = x - y;
}
return x;
}

}
```

Code Explanation:

1.rand(int min, int max, int count) method:

This method takes three integer parameters: min, max, and count.

It generates a random number between min and max based on the value of count.

If min is 1 and max is 6, it uses the remainder of count divided by 6 to determine the random number:

If the remainder is 0, it returns 3.

If the remainder is 1, it returns 2.

If the remainder is 2, it returns 5.

If the remainder is 3, it returns 1.

If the remainder is 4, it returns 6.

If the remainder is 5, it returns 4.

If min is not 1 or max is not 6, it uses the remainder of count divided by 7 to determine the random number using a similar logic.

If none of the conditions match, it returns 3.

2.Comparison methods:

The class defines three comparison methods: lt(int num1, int num2), gt(int num1, int num2), and eq(int num1, int num2).

These methods compare two integers (num1 and num2) and return a boolean value indicating the result of the comparison.

lt() returns true if num1 is less than num2, and false otherwise.

gt() returns true if num1 is greater than num2, and false otherwise.

eq() returns true if num1 is equal to num2, and false otherwise.

3.Other utility methods:

The class defines two additional utility methods: isEven(int num) and remainder(int x, int y).

isEven() takes an integer num as input and returns true if num is an even number (divisible by 2), and false otherwise.

remainder() calculates the remainder of dividing x by y using a while loop and repeatedly subtracting y from x until x is no longer greater than y.

The method then returns the value of x, which is the remainder of the division.

4.Overall, the RandomNumberGenerator class provides methods for generating random numbers based on certain conditions, performing integer comparisons, and calculating remainders. It can be used as a utility class in various applications that require random number generation or comparison operations.

3.2.3 Main

Code:

```
class Main{

function void main(){

   var String player_name;

   var char key;

   var int
count,ComputerWon,UserWon,Target,Score,ComputerThrew,UserThrew,UserW
onToss,TossResultOdd,PlayerToss,ComputerToss,TotalTossNumber,TossValue;
```

```
let count = 0;
let UserWon = 0;
let ComputerWon = 0;

let player_name = Keyboard.readLine("Enter the name of the player : ");
do ClearScreen.clearScreen();

do Output.moveCursor(10,30);
do Output.printString("HI ");
do Sys.wait(333);
do Output.printString(player_name);
do Output.printString(".");
do Output.println();

do Output.moveCursor(11,18);
do Sys.wait(333);
do Output.printString("Welcome to VIRTUAL CRICKET !");

do Output.moveCursor(15,26);
do Sys.wait(333);
do Output.printString("press any key");

let key = Keyboard.readChar();
if( key = 13 ){
do ClearScreen.clearScreen();
}
else{ do ClearScreen.clearScreen();}

do Output.println();
do Sys.wait(333);
do Output.printString("INSTRUCTIONS");
do Output.println();
do Output.printString("------------------------------------------------------------");
```

```
    do Output.println();
    do Output.printString("1. This game is a digital equilivalent of Hand Cricket.");
    do Output.println();
    do Output.printString("2. You play against Computer.");
    do Output.println();
    do Output.printString("3. There's toss, then Inning 1 and 2, just like a normal
game of cricket.");
    do Output.println();
    do Output.printString("4. But, You play the toss to bat. Whoever wins the toss,
Bats first.");
    do Output.println();
    do Output.printString("5. Every time you and computer throw a number.");
    do Output.println();
    do Output.printString("6. If the numbers match, the batsman is OUT.");
    do Output.println();
    do Output.printString("7. Else the batsman scores the number they throw.");
    do Output.println();
    do Output.printString("8. 0 is a defense for batsman, the bowler cannot throw a
0.");
    do Output.println();
    do Output.printString("9. Concept of Multiple Wickets and Overs don't
apply.");
    do Output.println();
    do Output.printString("10. The concept of score and target is the same as a
normal game of cricket.");
    do Output.println();
    do Output.println();
    do Output.printString("That's it! Enjoy the game! ");
    do Output.println();
    do Output.printString("----------------------------------------------------------------");
    do Output.println();


    do Sys.wait(333);
    do Output.printString("press enter");
    let key = Keyboard.readInt("");
```

```
do ClearScreen.clearScreen();

do Output.printString("TOSS TIME");
do Output.println();
do Output.printString("------------------------------------------------------------");

do Sys.wait(333);
let TossValue = Keyboard.readInt("Choose ODD or EVEN (1/0) : ");
do Output.println();
do Output.printString("------------------------------------------------------------");

if (TossValue = 1){
do Sys.wait(333);
do Output.printString("You have choosen ODD");
do Output.println();
do Sys.wait(333);
do Output.printString("Computer's call is EVEN");
do Output.println();
do Output.printString("------------------------------------------------------------");
}

else {if (TossValue = 0){
do Sys.wait(333);
do Output.printString("You have choosen EVEN");
do Output.println();
do Sys.wait(333);
do Output.printString("Computer's call is ODD");
do Output.println();
do Output.printString("------------------------------------------------------------");
}
else {
do Sys.wait(333);
let TossValue = Keyboard.readInt("Toss value can only be 1 or 0 : ");
```

```
    do Output.println();
    }
    }

    do Sys.wait(333);
    let PlayerToss = Keyboard.readInt("Throw toss (a number between 0 and 6) :
");
    do Output.println();
    let ComputerToss = RandomNumberGenerator.rand(1,6,count);
    let count = count + 1;

    if(RandomNumberGenerator.lt(PlayerToss,0) |
RandomNumberGenerator.gt(PlayerToss,6)){
    let PlayerToss = Keyboard.readInt("toss can only be a number between 0 and 6,
Re-enter : ");
    }

    do Output.printString(player_name);
    do Output.printString(" threw : ");
    do Output.printInt(PlayerToss);
    do Output.println();

    do Sys.wait(333);
    do Output.printString("COMPUTER threw : ");
    do Output.printInt(ComputerToss);
    do Output.println();

    do Sys.wait(333);
    do Output.printInt(PlayerToss);
    do Output.printString(" + ");
    do Output.printInt(ComputerToss);
    do Output.printString(" = ");
    let TotalTossNumber = PlayerToss + ComputerToss;
    do Output.printInt(TotalTossNumber);
```

```
do Output.println();

do Sys.wait(333);
if(RandomNumberGenerator.isEven(TotalTossNumber)){
do Output.printInt(TotalTossNumber);
do Output.printString(" is EVEN.");
do Output.println();
let TossResultOdd = 0;
}
else {
do Sys.wait(333);
do Output.printInt(TotalTossNumber);
do Output.printString(" is ODD.");
do Output.println();
let TossResultOdd = 1;
}

do Sys.wait(333);
do Output.printString("------------------------------------------------------------");
if(RandomNumberGenerator.eq(TossValue,TossResultOdd)){
do Output.printString("Congratulations, ");
do Output.printString(player_name);
do Output.printString(" WON THE TOSS!");
do Output.println();
do Sys.wait(333);
do Output.printString(player_name);
do Output.printString(" BATS FIRST.");
let UserWonToss = 1;
}
else{
do Sys.wait(333);
do Output.printString("Oops, COMPUTER WON THE TOSS!");
do Output.println();
```

```
do Sys.wait(333);
do Output.printString("COMPUTER BATS FIRST");
let UserWonToss = 0;
}

do Output.println();
do Output.println();
do Sys.wait(333);
do Output.printString("press enter");
let key = Keyboard.readInt("");
do ClearScreen.clearScreen();

if (UserWonToss=1){
do Output.moveCursor(0,0);
do Sys.wait(333);
do Output.printString("1st INNINGS");
do Output.println();
do Output.printString("----------------------------------------------------------------");
do Sys.wait(333);
do Output.printString(player_name);
do Output.printString(" : BATTING");
do Output.println();
do Sys.wait(333);
do Output.printString("COMPUTER : BOWLING");
do Output.println();
do Output.printString("----------------------------------------------------------------");

let UserThrew = 0;
let ComputerThrew = 1;
let Score = 0;
let Target = 1;

while (~(RandomNumberGenerator.eq(UserThrew,ComputerThrew))){
```

```
    do Sys.wait(333);
    do Output.printString(player_name);
    let UserThrew = Keyboard.readInt(" threw (Enter a number between 0 and 6) :
");
    if(RandomNumberGenerator.lt(UserThrew,0) |
RandomNumberGenerator.gt(UserThrew,6)){
    let UserThrew = Keyboard.readInt("The number should be in between 0 and 6,
Re-enter : ");
    }
    let ComputerThrew = RandomNumberGenerator.rand(1,6,count);
    let count = count + 1;
    do Sys.wait(333);
    do Output.printString("COMPUTER threw : ");
    do Output.printInt(ComputerThrew);
    do Output.println();

    if (RandomNumberGenerator.eq(UserThrew,ComputerThrew)){
    do Output.println();
    do Sys.wait(333);
    do Output.printString("THAT's OUT!");
    do Output.println();
    do Sys.wait(333);
    do Output.printString("Total 1st innings Score : ");
    do Output.printInt(Score);
    do Output.println();
    do Sys.wait(333);
    do Output.printString("Computer need ");
    do Output.printInt(Target);
    do Output.printString(" runs to win.");
    do Output.println();
    do Output.printString("------------------------------------------------------------");
    do Output.printString("                                                  ");
    do Output.printString("                                                  ");
```

```
do Sys.wait(333);
do Output.printString("press enter to start 2nd innings");
let key = Keyboard.readInt("");
do ClearScreen.clearScreen();
}

else{
let Score = Score + UserThrew;
let Target = Score + 1;
do Sys.wait(333);
do Output.printString(player_name);
do Output.printString(" scored ");
do Output.printInt(UserThrew);
do Output.printString(" runs in this ball.");
do Output.println();
do Sys.wait(333);
do Output.printString("Total 1st innings score : ");
do Output.printInt(Score);
do Output.println();
do Output.printString("----------------------------------------------------------------");
do Output.println();
do Output.printString("press enter to play the next ball");
let key = Keyboard.readInt("");
do Screen.clearScreen();
}
}// while loop dhi

do Sys.wait(333);
do Output.printString("2nd INNINGS");
do Output.println();
do Output.printString("----------------------------------------------------------------");
do Sys.wait(333);
do Output.printString(player_name);
```

```
    do Output.printString(" : BOWLING");
    do Output.println();
    do Sys.wait(333);
    do Output.printString("COMPUTER : BATTING");
    do Output.println();
    do Output.printString("------------------------------------------------------------");

    let UserThrew = 1;
    let ComputerThrew = 0;
    let Score = 0;

    while ((RandomNumberGenerator.lt(Score,Target))&(UserWon = 0)){

    do Sys.wait(333);
    do Output.printString(player_name);
    let UserThrew = Keyboard.readInt(" threw (Enter a number between 1 and 6) :
");
    if(RandomNumberGenerator.lt(UserThrew,1) |
RandomNumberGenerator.gt(UserThrew,6)){
    let UserThrew = Keyboard.readInt("The number should be in between 1 and 6,
Re-enter : ");
    }
    do Output.println();
    let ComputerThrew = RandomNumberGenerator.rand(0,6,count);
    let count = count + 1;
    do Sys.wait(333);
    do Output.printString("COMPUTER threw : ");
    do Output.printInt(ComputerThrew);
    do Output.println();

    if (RandomNumberGenerator.eq(UserThrew,ComputerThrew)){
    let UserWon = 1;
    do Output.println();
    do Sys.wait(333);
```

```
do Output.printString("THAT's OUT!");
do Output.println();
do Output.println();
do Sys.wait(333);
do Output.printString("press enter");
let key = Keyboard.readInt("");
do ClearScreen.clearScreen();
do Output.printString("----------------------------------------------------------------");
do Output.println();
do Sys.wait(333);
do Output.printString("CONGRATULATIONS ");
do Sys.wait(333);
do Output.printString(player_name);
do Sys.wait(333);
do Output.printString(". YOU WON !!!");
do Output.println();
do Output.println();
do Output.printString("----------------------------------------------------------------");
}

else{
let Score = Score + ComputerThrew;
do Sys.wait(333);
do Output.printString("Computer scored ");
do Output.printInt(ComputerThrew);
do Output.printString(" runs in this ball.");
do Output.println();
do Sys.wait(333);
do Output.printString("Total runs chased : ");
do Output.printInt(Score);
do Output.println();
do Sys.wait(333);
do Output.printString("To win : ");
```

```
do Output.printInt(Target-Score);
do Output.println();
do Output.printString("------------------------------------------------------------");
do Output.println();
do Output.printString("press enter to play the next ball");
let key = Keyboard.readInt("");
do Screen.clearScreen();
}
}// while loop dhi

if(RandomNumberGenerator.gt(Score,Target)){
do Sys.wait(333);
do Output.printString("press enter");
let key = Keyboard.readInt("");
do ClearScreen.clearScreen();
do Output.printString("------------------------------------------------------------");
do Output.println();
do Sys.wait(333);
do Output.printString("OOPS ! COMPUTER WON.");
do Output.println();
do Sys.wait(333);
do Output.printString("AI tho pettukune gallanthey bradhaaarrrr.. It states :)");
do Output.println();
do Output.println();
do Output.printString("------------------------------------------------------------");
}
}//idh toss user gelisthe vachhina loop ni end chesedhi
else{
do Output.moveCursor(0,0);
do Sys.wait(333);
do Output.printString("1st INNINGS");
do Output.println();
do Output.printString("------------------------------------------------------------");
```

```
    do Sys.wait(333);
    do Output.printString(player_name);
    do Output.printString(" : BOWLING");
    do Output.println();
    do Sys.wait(333);
    do Output.printString("COMPUTER : BATTING");
    do Output.println();
    do Output.printString("------------------------------------------------------------");

    let UserThrew = 1;
    let ComputerThrew = 0;
    let Score = 0;
    let Target = 1;

    while (~(RandomNumberGenerator.eq(UserThrew,ComputerThrew))){

    do Sys.wait(333);
    do Output.printString(player_name);
    let UserThrew = Keyboard.readInt(" threw (Enter a number between 1 and 6) :
");
    if(RandomNumberGenerator.lt(UserThrew,1) |
RandomNumberGenerator.gt(UserThrew,6)){
    let UserThrew = Keyboard.readInt("The number should be in between 1 and 6,
Re-enter : ");
    }
    let ComputerThrew = RandomNumberGenerator.rand(0,6,count);
    let count = count + 1;
    do Sys.wait(333);
    do Output.printString("COMPUTER threw : ");
    do Output.printInt(ComputerThrew);
    do Output.println();

    if (RandomNumberGenerator.eq(UserThrew,ComputerThrew)){
    do Output.println();
```

```
do Sys.wait(333);
do Output.printString("THAT's OUT!");
do Output.println();
do Sys.wait(333);
do Output.printString("Total 1st innings Score : ");
do Output.printInt(Score);
do Output.println();
do Sys.wait(333);
do Output.printString(player_name);
do Output.printString(" need ");
do Output.printInt(Target);
do Output.printString(" runs to win.");
do Output.println();
do Output.printString("------------------------------------------------------------");
do Output.printString("                                                  ");
do Output.printString("                                                  ");
do Sys.wait(333);
do Output.printString("press enter to start 2nd innings");
let key = Keyboard.readInt("");
do ClearScreen.clearScreen();
}

else{
let Score = Score + ComputerThrew;
let Target = Score + 1;
do Sys.wait(333);
do Output.printString("Computer scored ");
do Output.printInt(ComputerThrew);
do Output.printString(" runs in this ball.");
do Output.println();
do Sys.wait(333);
do Output.printString("Total 1st innings score : ");
do Output.printInt(Score);
```

```
    do Output.println();
    do Output.printString("------------------------------------------------------------");
    do Output.println();
    do Output.printString("press enter to play the next ball");
    let key = Keyboard.readInt("");
    do Screen.clearScreen();
    }
    }// while loop dhi

    do Sys.wait(333);
    do Output.printString("2nd INNINGS");
    do Output.println();
    do Output.printString("------------------------------------------------------------");
    do Sys.wait(333);
    do Output.printString(player_name);
    do Output.printString(" : BATTING");
    do Output.println();
    do Sys.wait(333);
    do Output.printString("COMPUTER : BOWLING");
    do Output.println();
    do Output.printString("------------------------------------------------------------");

    let UserThrew = 0;
    let ComputerThrew = 1;
    let Score = 0;

    while ((RandomNumberGenerator.lt(Score,Target))&(ComputerWon = 0)){

    do Sys.wait(333);
    do Output.printString(player_name);
    let UserThrew = Keyboard.readInt(" threw (Enter a number between 0 and 6) :
");
    if(RandomNumberGenerator.lt(UserThrew,0) |
RandomNumberGenerator.gt(UserThrew,6)){
```

```
   let UserThrew = Keyboard.readInt("The number should be in between 0 and 6,
Re-enter : ");
   }
   let ComputerThrew = RandomNumberGenerator.rand(1,6,count);
   let count = count + 1;
   do Sys.wait(333);
   do Output.printString("COMPUTER threw : ");
   do Output.printInt(ComputerThrew);
   do Output.println();

   if (RandomNumberGenerator.eq(UserThrew,ComputerThrew)){
   let ComputerWon = 1;
   do Output.println();
   do Sys.wait(333);
   do Output.printString("THAT's OUT!");
   do Output.println();
   do Output.println();
   do Sys.wait(333);
   do Output.printString("press enter");
   let key = Keyboard.readInt("");
   do ClearScreen.clearScreen();
   do Output.printString("------------------------------------------------------------");
   do Output.println();
   do Sys.wait(333);
   do Output.printString("OOPS ! COMPUTER WON.");
   do Output.println();
   do Sys.wait(333);
   do Output.printString("AI tho pettukune gallanthey bradhaaarrrr.. It states :)");
   do Output.println();
   do Output.println();
   do Output.printString("------------------------------------------------------------");
   }
```

```
else{
let Score = Score + UserThrew;
do Sys.wait(333);
do Output.printString(player_name);
do Output.printString(" scored ");
do Output.printInt(UserThrew);
do Output.printString(" runs in this ball.");
do Output.println();
do Sys.wait(333);
do Output.printString("Total runs chased : ");
do Output.printInt(Score);
do Output.println();
do Sys.wait(333);
do Output.printString("To win : ");
do Output.printInt(Target-Score);
do Output.println();
do Output.printString("----------------------------------------------------------------");
do Output.println();
do Output.printString("press enter to play the next ball");
let key = Keyboard.readInt("");
do Screen.clearScreen();
}
}// while loop dhi

if(RandomNumberGenerator.gt(Score,Target)){
do Sys.wait(333);
do Output.printString("press enter");
let key = Keyboard.readInt("");
do ClearScreen.clearScreen();
do Output.printString("----------------------------------------------------------------");
do Output.println();
do Sys.wait(333);
do Output.printString("CONGRATULATIONS ");
```

```
    do Sys.wait(333);
    do Output.printString(player_name);
    do Sys.wait(333);
    do Output.printString(". YOU WON !!!");
    do Output.println();
    do Output.println();
    do Output.printString("----------------------------------------------------------------");
    }
    }//idhi toss computer gelisthe vachhina loop
    return;
    }
}
```

Code Explanation:

1.The code defines a class named "Main" that contains a function named "main".

2.Inside the "main" function, several variables are declared, including "player_name", "key", "count", "ComputerWon", "UserWon", "Target", "Score", "ComputerThrew", "UserThrew", "UserWonToss", "TossResultOdd", "PlayerToss", "ComputerToss", "TotalTossNumber", and "TossValue". These variables are used to store various game-related information.

3.The code uses keyboard input and output functions to interact with the user. For example, the "Keyboard.readLine" function is used to read the player's name, and "Keyboard.readChar" and "Keyboard.readInt" are used to read a single character and integer input, respectively.

4.The code clears the screen, prints some messages, and waits for the user to press a key to proceed.

5.The code provides instructions for playing the game, explaining the rules and concepts.

6.The code asks the player to choose odd or even for the toss and validates the input.

7.The code asks the player to throw a toss (a number between 0 and 6) and generates a random toss for the computer. It compares the toss values, determines odd/even, and displays the result of the toss.

8.Based on the toss result, the code determines whether the player or the computer wins the toss and assigns a value to the "UserWonToss" variable accordingly.

9.If the player wins the toss, the code enters the first innings where the player bats and the computer bowls. It uses a while loop to simulate the balls being played.

The player and computer throw numbers between 0 and 6, and the code checks if the numbers match or not. If the numbers match, the batsman is out; otherwise, the batsman scores the number they throw. The score is displayed after each ball.

10.If the player is out, the code calculates the target score for the second innings, clears the screen, and waits for the user to press enter to start the second innings.

11.In the second innings, the player bowls, and the computer bats. Similar to the first innings, the code simulates the balls being played, checks for matches, and updates the score.

12.At the end of the second innings, the code checks if the player's score is greater than the target score. If yes, the player wins; otherwise, the computer wins.

13.The code displays the result of the game and waits for the user to press enter to exit the game.

14.Overall, the code implements a simple text-based virtual cricket game where the player plays against the computer. It involves tossing, batting, and bowling to simulate a cricket match.

3.3EXPERIMENTS & RESULTS:

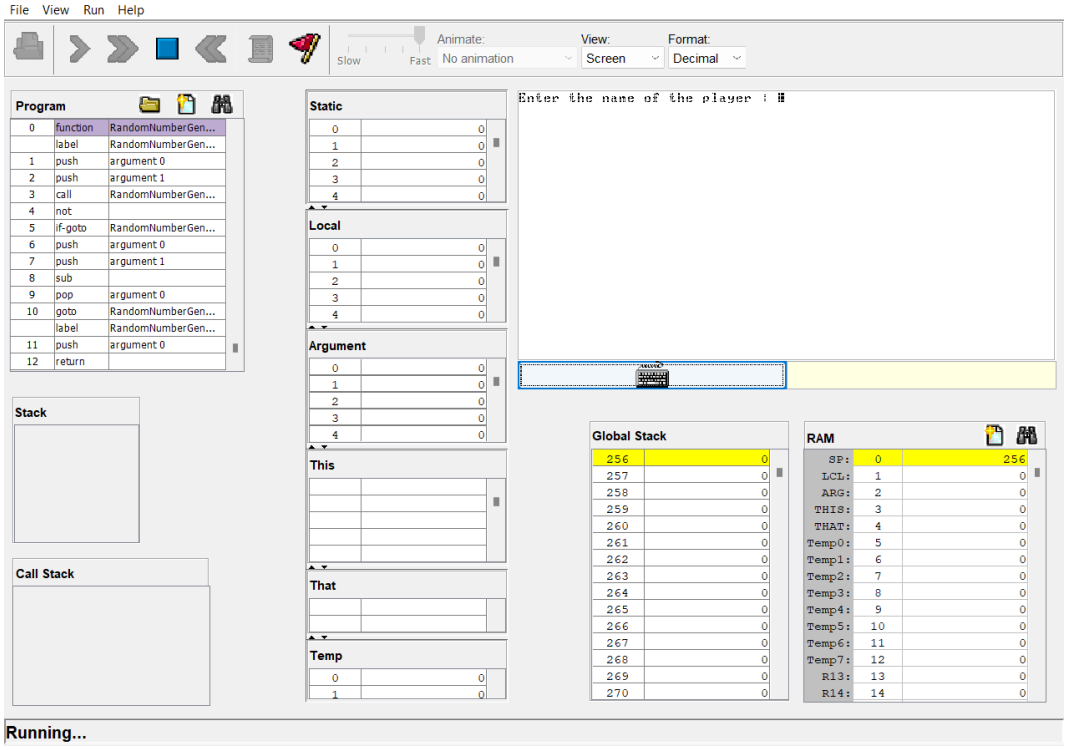


Figure 11 Part C output-1

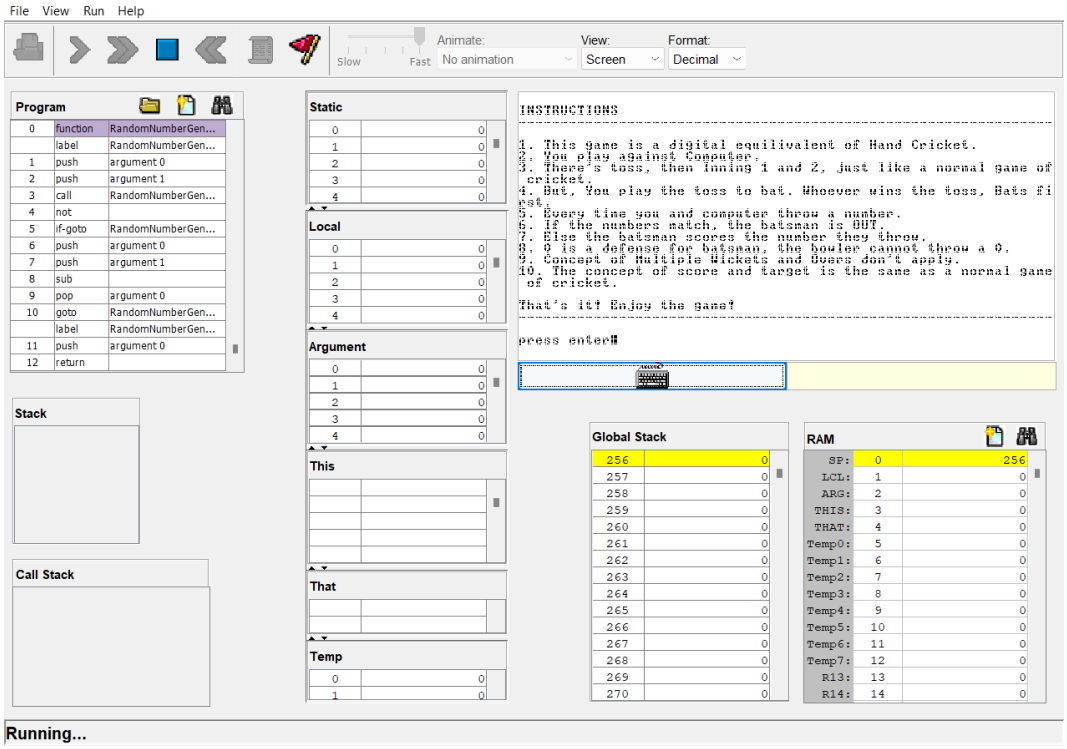


Figure 12 Part C output-2

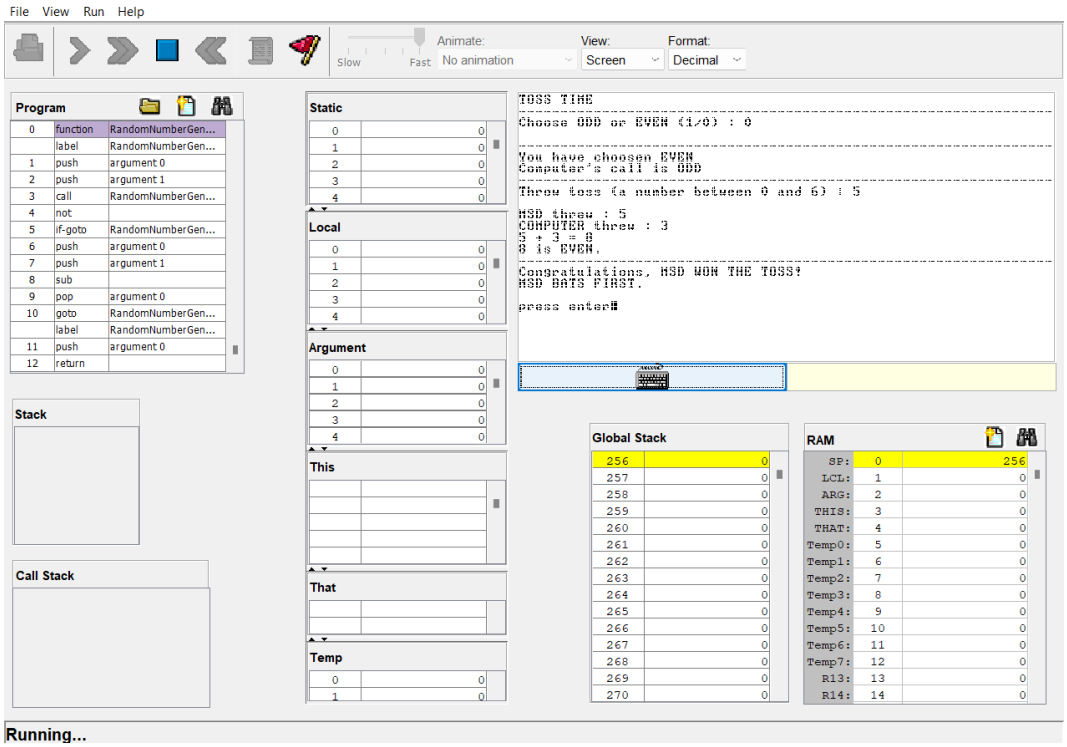


Figure 13 Part C output-3

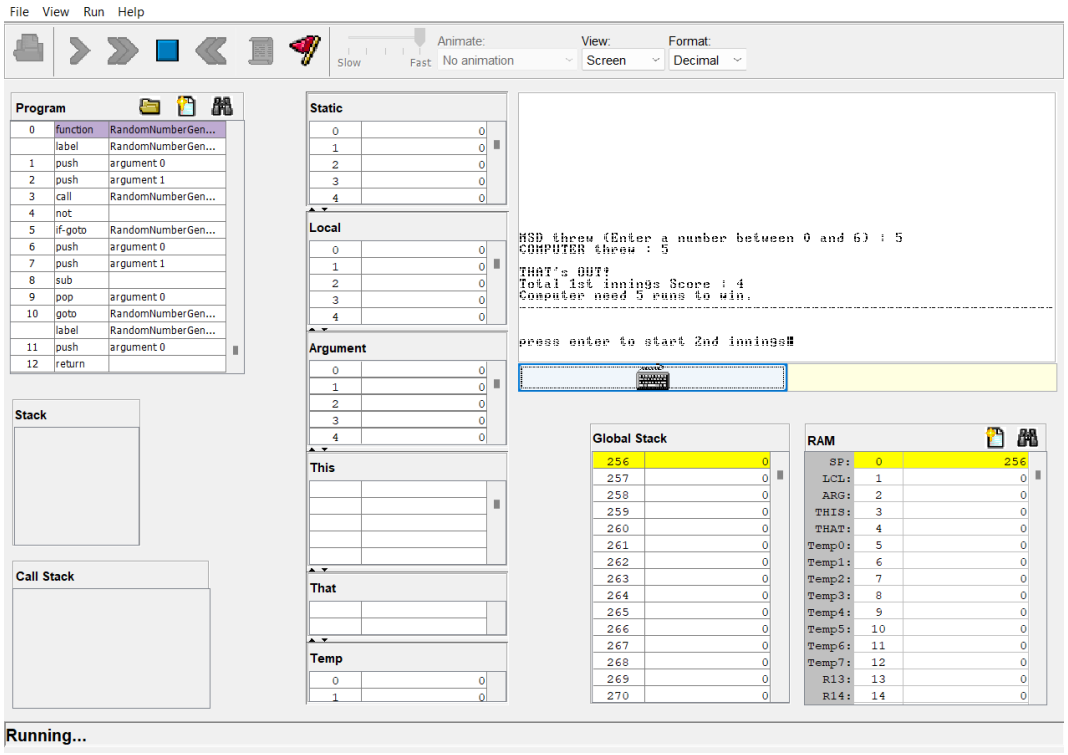


Figure 14 Part C output-4

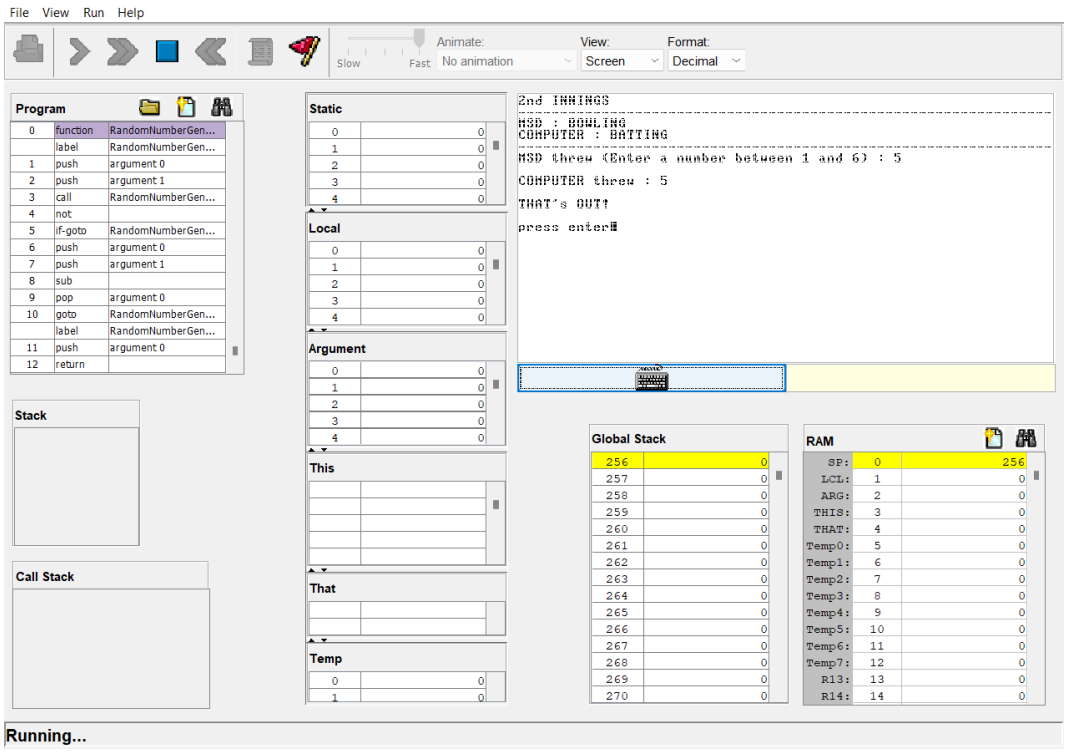


Figure 15 Part C output-5

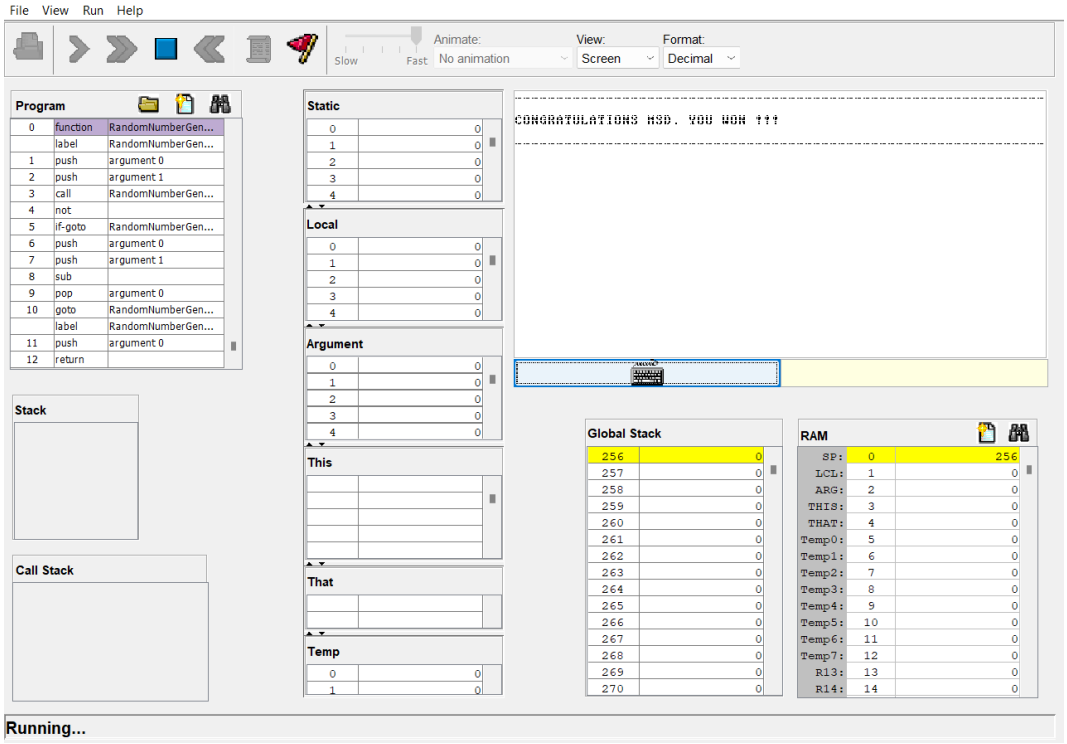


Figure 16 Part C output-6

CONCLUSION

Part A :

The computer system implemented in the Elements of Computing - 2 course can be concluded as a successful integration of three main components: Memory.hdl, CPU.hdl, and ROM32K. The Memory.hdl chip was responsible for combining RAM16K, Screen memory map, and keyboard memory map, enabling the storage of variables, objects, and input/output control. Although this report does not delve into the implementation details of CPU.hdl, it was developed separately in a previous semester. Serving as the storage for sequential execution instructions, the ROM32K chip played a crucial role in the overall functionality of the computer system. By interconnecting these three chips, a fully functional computer system was constructed.

The success of the computer implementation was further validated through the utilization of a provided test script. The script involved summing numbers from 10 to 50 and comparing the resulting output to the expected results. This comprehensive testing procedure served as a confirmation of the computer's functionality and showcased its ability to perform complex calculations accurately. Overall, the implementation of this computer system successfully demonstrated the integration of memory, processing, and instruction storage components to create a fully functional computing system.

In conclusion, the integration of the computer system in the Elements of Computing - 2 course marked a significant achievement. The successful combination of Memory.hdl, CPU.hdl, and ROM32K chips resulted in the creation of a functional computer system. The Memory.hdl chip provided storage capabilities for variables, objects, and input/output control, while the CPU.hdl chip, developed separately in a previous semester, handled the processing aspect. The ROM32K chip served as the storage for sequential execution instructions. The integration of these components resulted in the construction of a fully operational computer system. By conducting thorough testing, which included summing numbers and comparing outputs, the functionality of the system was verified. This project exemplified the successful integration of memory, processing, and instruction storage components, showcasing our understanding of computer architecture and implementation.

References:

ChatGPT

Part B :

In conclusion, our project successfully implemented a single-player Pong game using the Jack programming platform. The game consists of four main code files: Ball, Player, Game, and Main. The Ball file allows the user to select from three pre-existing ball designs and defines the movement of the ball. The Player file enables the user to choose the bat design and controls its movement. The Game file acts as the main while loop, ensuring the game functions as intended, based on the principles of the Pong game.

Throughout the development process, we encountered various challenges and overcame them through collaboration and problem-solving. We gained a deep understanding of game mechanics, object-oriented programming, and user interaction. Our project not only showcases our proficiency in the Jack programming language but also demonstrates our ability to design and implement an engaging and functional game.

Overall, the project has been a valuable learning experience, allowing us to apply our programming skills to create an entertaining and interactive game. Through this project, we have enhanced our understanding of software development methodologies, code organization, and debugging techniques. We are proud of our achievements and believe that our single-player Pong game in the Jack programming platform serves as a testament to our knowledge, creativity, and dedication.

References:

GitHub Repository: https://github.com/lucanevess87/PongGAME

ChatGPT

Part C :

In conclusion, our project successfully implemented a single-player Hand Cricket game against a computer opponent using the Jack programming platform. The project consists of three main code files: Clear Screen, RandomNumberGenerator, and Main. The Clear Screen file contains a function that clears the screen by printing a whole screen of empty spaces, ensuring a clean and user-friendly interface. The RandomNumberGenerator file encompasses multiple functions, with the main function being 'rand', which generates a random integer

between two specified arguments. Finally, the Main file contains the core code of the game, following the working methodology of Hand Cricket, where the player competes against the computer.

Throughout the development process, we encountered various challenges and effectively overcame them through collaborative problem-solving. By implementing the Hand Cricket game, we gained a deeper understanding of game logic, user interaction, and the utilization of random number generation. Our project not only demonstrates our proficiency in the Jack programming language but also showcases our ability to design and implement an engaging and interactive game.

Overall, this project has been a valuable learning experience, allowing us to apply our programming skills to create an enjoyable and challenging gaming experience. Through this project, we have enhanced our knowledge of software development methodologies, code organization, and user interface design. We are proud of our achievements and believe that our implementation of the single-player Hand Cricket game in the Jack programming platform serves as a testament to our creativity, problem-solving abilities, and dedication to producing high-quality software.

References:

ChatGPT