

# FINAL PROJECT

## ELEMENTS OF COMPUTING SYSTEMS - 02

### GROUP - 14

CB.ENU4AIE22017 - GANESH SUNDHAR S

CB.ENU4AIE22025 - KARTHIGAI SELVAM R

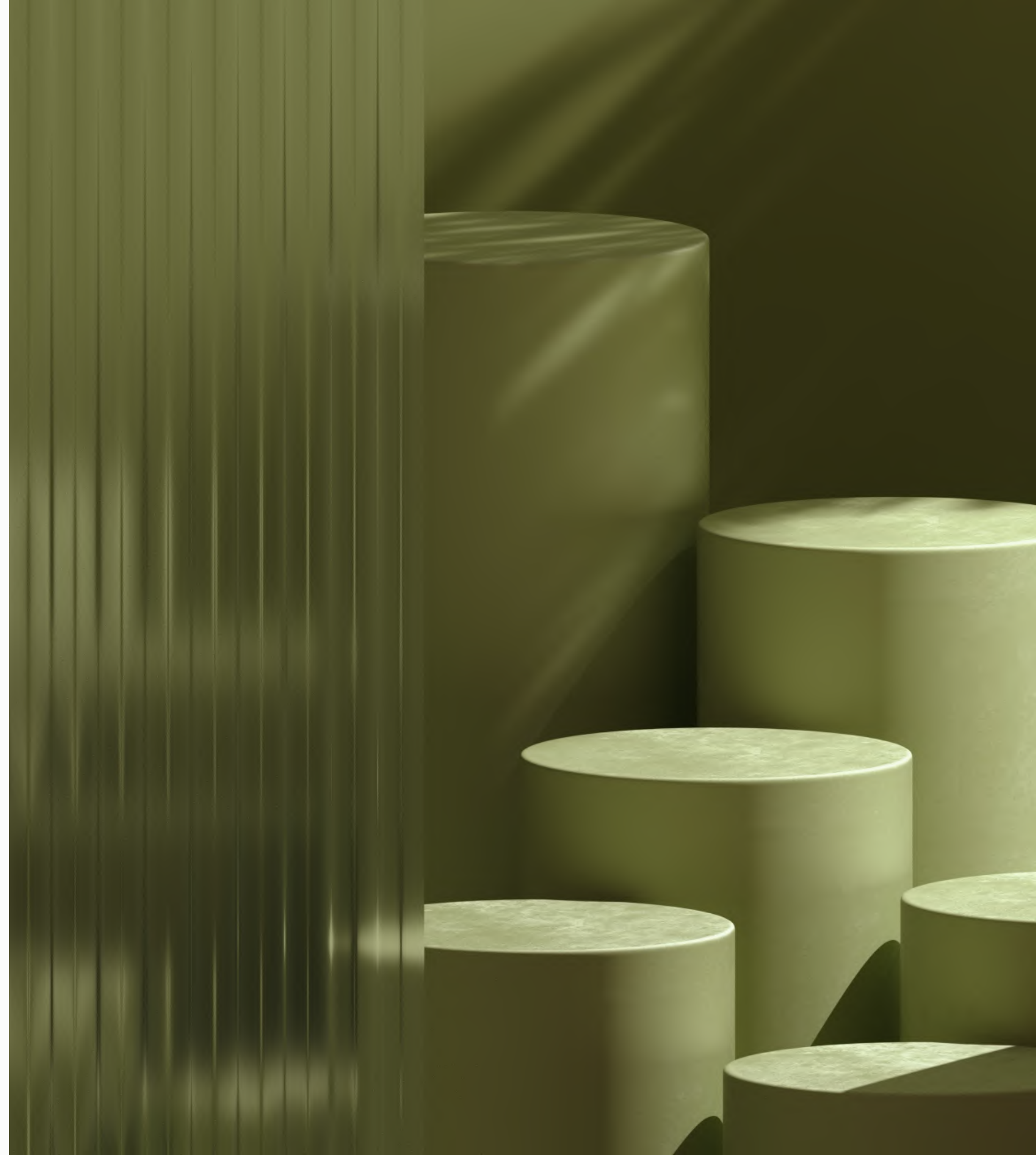
CB.EN.U4AIE22047 - SHRUTHIKAA V

CB.ENU4AIE22055 - Bhavya Sainath

# PRESENTATION TITLE

## AGENDA

- PART A - COMPUTER
- PART B - TIC TAC TOE
- PART C - FLAPPY BIRD AND ROCK PAPER AND  
SCISSOR GAME



# CONTRIBUTION

GANESH SUNDHAR S - PART B

KARTHIGAI SELVAM - PART A, PART B [Board], PART C [Pipes]

SHRUTHIKAA V - PART C [FLAPPY BIRD]

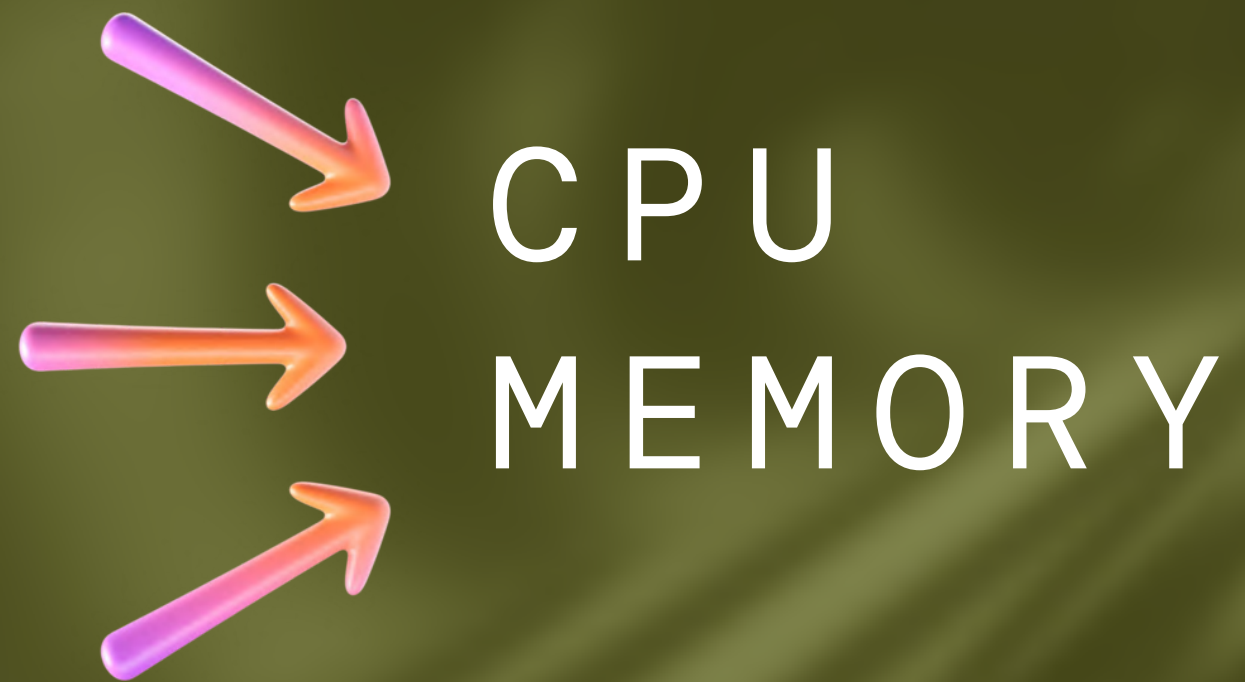
BHAVYA SAINATH - PART C [ROCK, PAPER AND SCISSOR]



# PART-A

## IMPLEMENT HACK COMPUTER

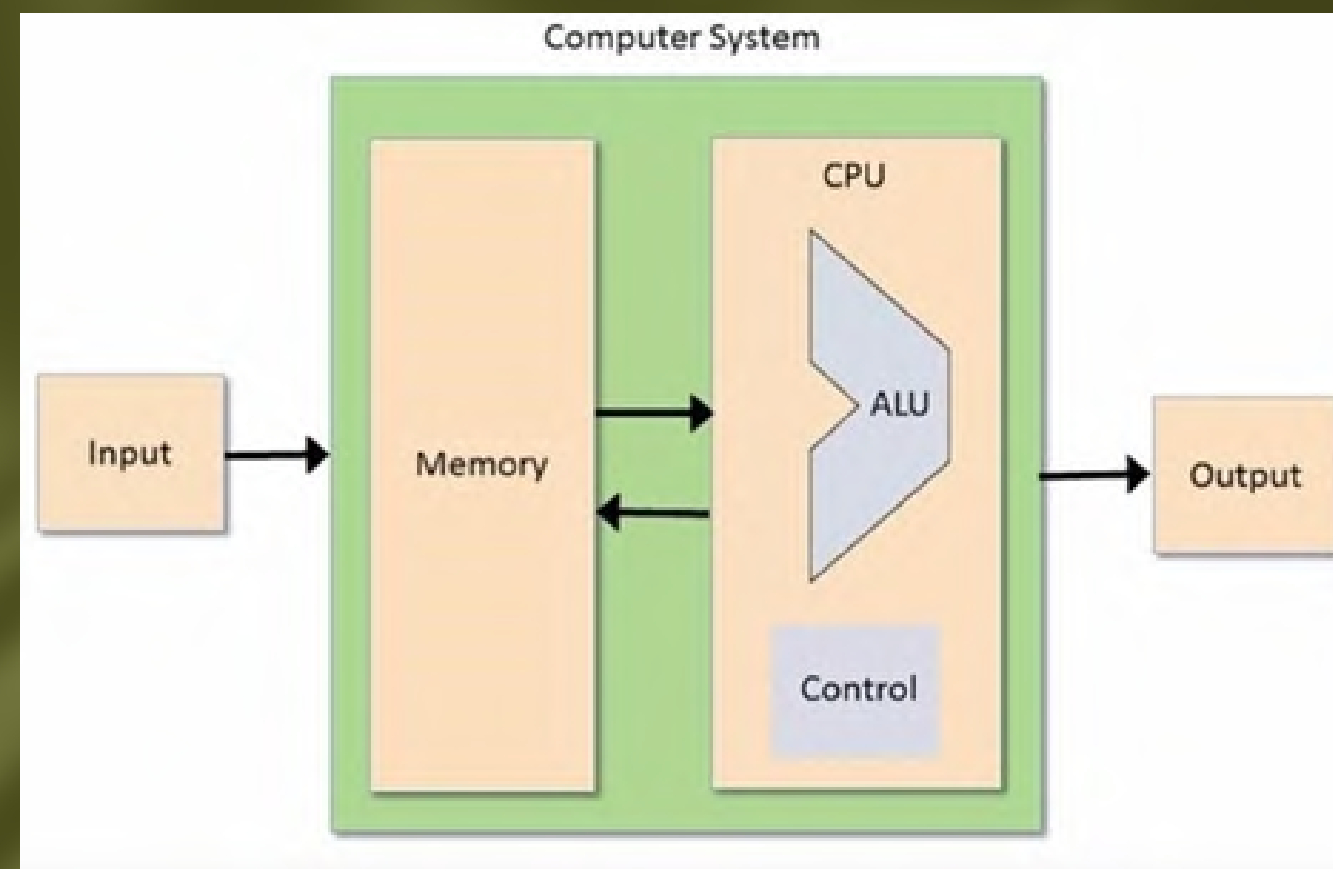
To build a HACK Computer, We need two components:



# Central Processing Unit (CPU):

*The CPU is the computer's central processing unit, and it is in charge of executing instructions and conducting computations. It is made up of a control unit, an ALU (Arithmetic Logic Unit), and registers.*

*CPU is made up of logic gates like Nand, Mux16, Register, ALU (Arithmetic Logic unit) and PC (Program Counter) logic gates where, ALU makes the function part and Program Counter works like a register that selected the load, reset, increment components and change the output accordingly*

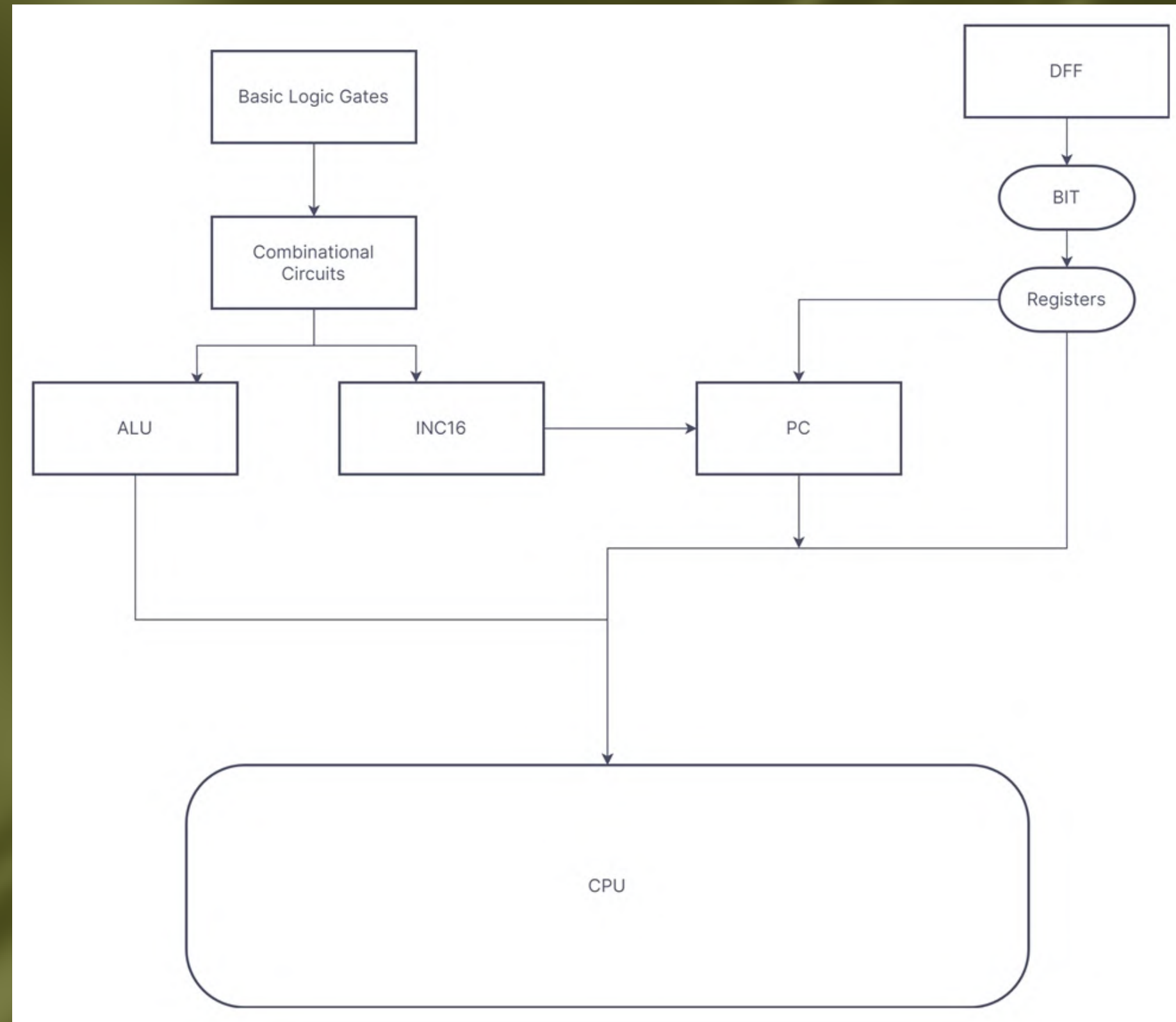




A CPU (Central Processing Unit) is the brain of a computer and is responsible for executing instructions and performing arithmetic and logical operations. The structure of a CPU typically contains the following components:

1. **The control unit** retrieves instructions from memory and decodes them in order to determine the operation to be executed. It also handles the overall execution of the programme and governs the flow of data and signals between various sections of the CPU.
2. The **ALU** performs arithmetic and logical operations like as addition, subtraction, comparison, and bitwise operations. These operations are carried out in accordance with the instructions received from the control unit.
3. **Registers:** Registers are compact, quick storage units that are used to store interim results and other data. Each register on the CPU serves a distinct purpose, such as the programme counter.

# FLOW CHART





# Memory:

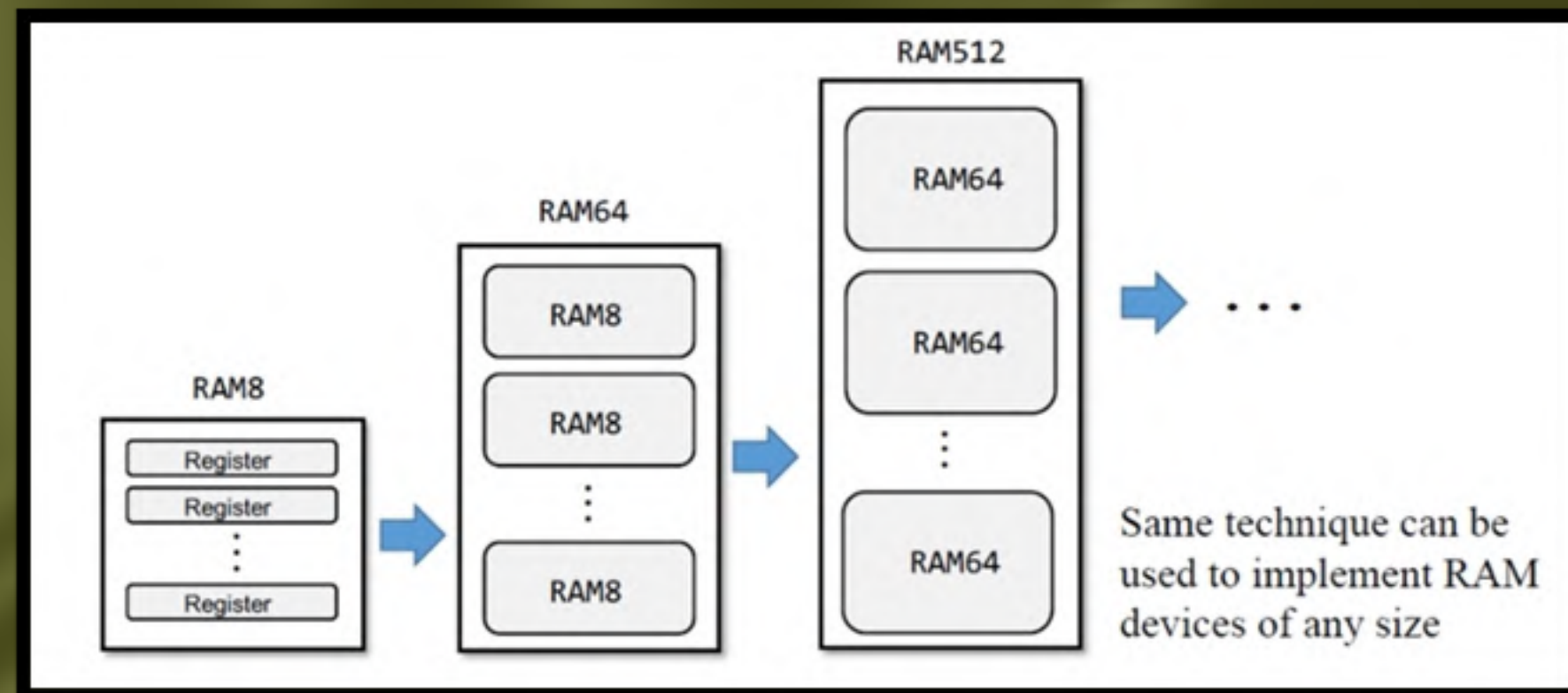
*Here's an overview of the memory components you'll typically encounter in the course:*

- 1. **Bit:** The smallest unit of memory in a computer, representing a single binary digit (0 or 1).*
- 2. **Register:** A group of sequential bits used to store and manipulate data within the computer. In the course, you'll typically work with 16-bit registers.*
- 3. **RAM (Random Access Memory):** RAM is a type of volatile memory that allows for both reading and writing data. In the context of the course, you'll design a RAM chip capable of storing multiple registers.*
- 4. **Memory Address:** Each memory location in RAM is assigned a unique address, which is used to identify and access specific data. In the case of the course, you'll typically work with a 15-bit memory address.*



5. **Memory Module:** A memory module combines multiple RAM chips to provide a larger storage capacity. In the course, you'll construct a memory module capable of storing 16K (16,384) 16-bit values.

6. **Memory Access:** The process of reading or writing data from/to a specific memory location. You'll design circuitry that allows the computer's CPU to access and interact with the memory module.



HDL is a specialized language used for describing and simulating digital circuits. Here's a high-level overview of the components you would typically need to implement in HDL:

## 1. Logic Gates:

The basic building blocks of digital circuits are logic gates. In the NAND2Tetris course, you start by implementing elementary logic gates, such as NAND, AND, OR, and XOR gates. These gates serve as the foundation for building more complex components.

## 2. Arithmetic Logic Unit (ALU):

The ALU is responsible for performing arithmetic and logical operations. It takes input from registers and performs operations like addition, subtraction, bitwise operations, and comparisons. You would need to write HDL code for the ALU, considering the specific operations and bit widths required by the computer architecture.

## 3. Registers:

Registers are used for storing data temporarily. In the Hack computer architecture, there are several types of registers, including the A register, D register, and memory registers (RAM). You would need to write HDL code to describe these registers and their functionalities.



## 4. Memory:

The Hack computer uses Random Access Memory (RAM) for storing instructions and data. The memory is organized into a large addressable space, and you would need to design and implement the memory module using HDL code. The course introduces the concept of a memory chip, which is essentially a collection of registers.

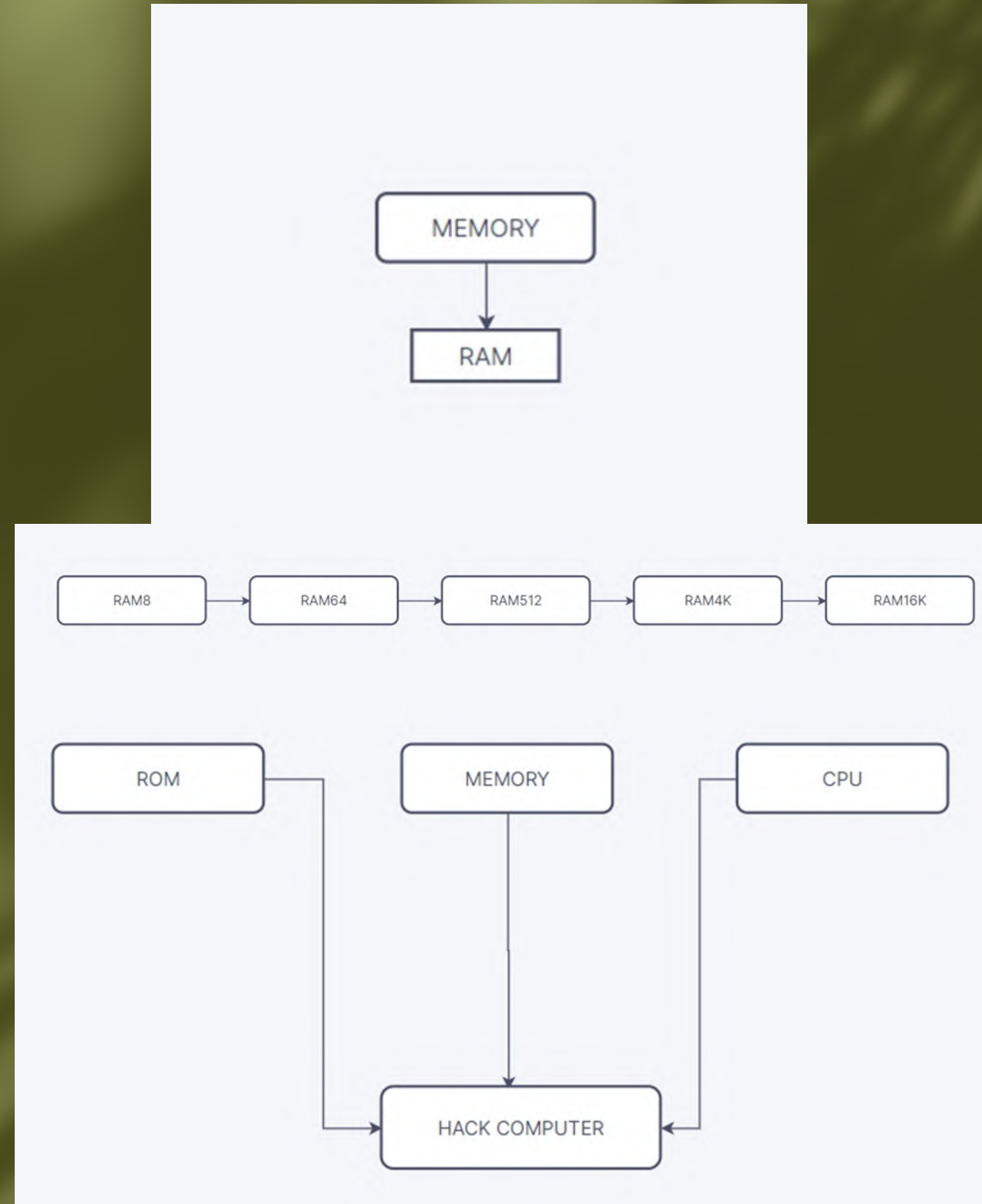
## 5. CPU:

The CPU (Central Processing Unit) is the heart of the computer. It fetches instructions, performs calculations, and controls the flow of data. To build the CPU, you would need to combine the ALU, registers, memory, and other components into a cohesive unit. The CPU executes instructions based on the architecture's instruction set, which includes A-instructions, C-instructions, and jump instructions.

## 6. Control Logic:

The control logic coordinates the operation of the CPU and ensures that the correct signals are generated for each instruction. It controls the flow of data, performs instruction decoding, and generates control signals to activate the appropriate components. Writing HDL code for the control logic involves implementing finite state machines and conditional logic to execute instructions.

# FLOW CHART














# TESTING

Hardware Simulator (2.5) - C:\Users\Karthi\Downloads\nand2tetris\nand2tetris\projects\05\Computer.hdl

File View Run Help



SlowFast

Animate:No animationFormat:DecimalView:Script

Chip Name :Computer (Clocked)Time :3021

Input pins

Name	Value
reset	0

Output pins

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

HDL

// This file is part of www.nar  
// and the book "The Elements c  
// by Nisan and Schocken, MIT E  
// File name: projects/05/Compu  
  
/\*\*  
 \* The HACK computer, including  
 \* When reset is 0, the program  
 \* When reset is 1, the executi  
 \* Thus, to start a program's e  
 \* and "down" (0). From this pc  
 \* the software. In particular,  
 \* screen may show some output  
 \* with the computer via the ke

Internal pins

Name	Value
pc[15]	256
instruction[16]	17
inM[16]	256
load	0
outM[16]	0
address[15]	2

```
load Computer.hdl,  
compare-to Computer.cmp;  
output-file Computer.out,  
output-list time%S1.4.1 reset%B2.1.2 ARegister[0]%D1.7.1 DRegister[0]%D  
ROM32K load factorial.hack,  
output;  
repeat 3021{  
    tick, tock, output;  
}
```

End of script - Comparison ended successfully

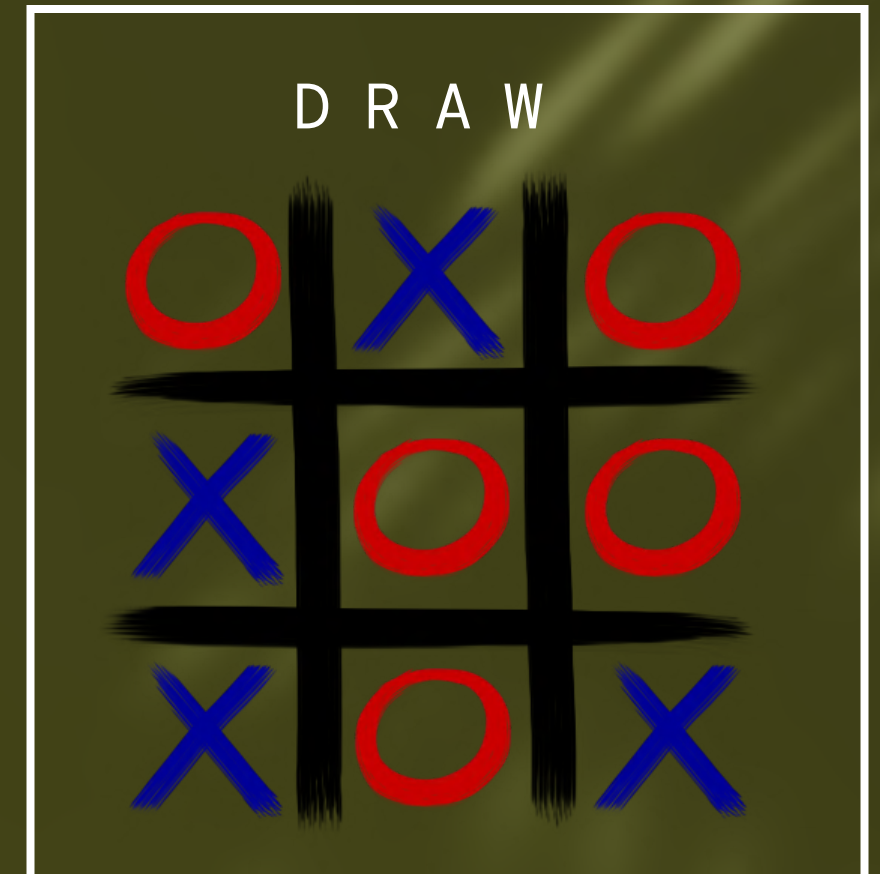
# PART B

## TIC TAC TOE

### THE CORE IDEA

Tic tac toe is a game played by two or more people. The game consists of a 3 x 3 board. The main objective of the game is to place the marks of a single player along either on one of the rows or columns or along one of the diagonals.

The game can be played with either with the computer or with another player near you.





# PART B

## MAIN CLASS

The main class is used to declare the main function as Main.main is the entry point of any jack program. Other classes and functions can be called only from the main function.

From the main function, we create a variable of the Game class. Then we call the go function from that class using that object. Then we declare the return statement of the main function. Then we close the main function and the Main class.

```
class Main{  
    function void main(){  
        var Game g;  
        do g.go();  
        return;    }  
}
```

# PART B

## GAME CLASS

We declare the variables that we use initially. This is done because we can't declare variables in the middle of any jack code.

Then, we do clear screen initially in order to clean the screen before we print any statements. Then, we mark the documentation part in our program by calling the documentation function. Then, we are going to draw the grid by calling the draw function. The clear screen function is available in the Screen OS Class. The documentation and draw functions are available in the Setup class from our program.

```
var Setup set;  
var int check;  
var Single single;  
var Double double;  
  
do Screen.clearScreen();  
do set.documentation();  
do set.draw();  
let check = set.player_computer();
```



# PART B

## SETUP CLASS

```
method void documentation(){  
    do Output.moveCursor(22,6);  
    do Output.printString("Elements of Computing Systems - 2 : Group 14 | PART B");  
    do Output.moveCursor(0,0);  
    return;  
}
```

```
method int player_computer(){  
    var int check;  
    let check = Keyboard.readInt("Single Player(1) | Double Player(2) : ");  
    return check;  
}
```

# PART B

## SETUP CLASS

```
method void draw(){  
    do Screen.setColor(true);  
    do Screen.drawRectangle(156,82,356,92);  
    do Screen.drawRectangle(156,152,356,162);  
    do Screen.drawRectangle(216,22,226,222);  
    do Screen.drawRectangle(286,22,296,222);  
    return;  
}
```



# PART B

## GAME CLASS

```
if(check = 1){
    do set.clear_line();
    do single.run();
}
else{
    if(check = 2){
        do set.clear_line();
        do double.run();
    }
    else{
        do set.clear_line();
        do Output.printString("ERROR : Please enter either 1 or 2");
        do Sys.wait(2000);
        do go();
    }
}
```

This piece of code checks if the value entered is either 1 or 2 and then executes the clear line function. Also, it calls the run function of the respective classes. It also prints an error if the value is not 1 or 2.

# PART B

## GAME CLASS

```
do Sys.wait(3500);  
do Screen.clearScreen();  
do Output.moveCursor(11,17);  
do Output.printString("CLICK 'R' TO RESTART THE GAME");  
while(~(Keyboard.keyPressed() = 82)){  
}  
do go();  
return;
```

This code is for calling the function again whenever 'R' is pressed. This works as a restart for the game while in action.

82 is the ASCII value of 'R'



# PART B

## SINGLE CLASS

```
var Array row1,row2,row3;
var Array board1;
var boolean Game_Over,win1,win2;
var int player,location,row,col,i,j,k,computer,temp;
var Setup set;
let Game_Over = false;
let player = 1;
let row1 = Array.new(3);
let row2 = Array.new(3);
let row3 = Array.new(3);
let board1 = Array.new(3);
```

```
let row1[0] = 0;
let row1[1] = 0;
let row1[2] = 0;
let row2[0] = 0;
let row2[1] = 0;
let row2[2] = 0;
let row3[0] = 0;
let row3[1] = 0;
let row3[2] = 0;
let board1[0] = row1;
let board1[1] = row2;
let board1[2] = row3;

let computer = get();
let player = 1;
```

# PART B

## SINGLE CLASS

```
while(~(Game_Over)){
    if(player = computer){
        let location = move(board1,player,computer);
        if(player = 1){
            do set.draw_x(location);
        }
        else{
            do set.draw_o(location);
        }
        let location = location-1;
        let row = location/3;
        let col = location - (3 * (location/3));
        let board1 = set.set_board(board1,row,col,player);
        let Game_Over = set.check(board1,player);
        if(player = 1){
            let player = 2;
        }
        else{
            let player = 1;
        }
    }
}
```

```
else{
    let location = Keyboard.readInt("Your turn
(Enter a number between 1 to 9) : ");
    do set.clear_line();
    if((location<10) & (location>0)){
        let location = location-1;
        let row = location/3;
        let col = location - (3 * (location/3));
```



# PART B

## SINGLE CLASS

```
if(set.board(board1,row,col) = 0){
    let location = location+1;
    let board1 = set.set_board(board1,row,col,player);
    if(player = 1){
        do set.draw_x(location);
    }
    else{
        do set.draw_o(location);
    }
    let Game_Over = set.check(board1,player);
    if(player = 1){
        let player = 2;
    }
    else{
        let player = 1;
    }
}
```

```
    else{
        do Output.printString("ERROR :
The entered location is already used");
        do Sys.wait(2000);
        do set.clear_line();
    }
}
else{
    do Output.printString("ERROR :
Enter a valid location");
    do Sys.wait(2000);
    do set.clear_line();
}
```

# PART B

## SETUP CLASS

```
method void clear_line(){
    do Screen.setColor(false);
    do Screen.drawRectangle(0,0,511,10);
    do Output.moveCursor(0,0);
    return;
}
```

```
method int board(Array arr,int x,int y){
    var int temp;
    var Array temp1;
    let temp1 = arr[x];
    let temp = temp1[y];
    return temp;
}
```

```
method Array set_board(Array arr,int x,int y,int val){
    var Array temp;
    let temp = arr[x];
    let temp[y] = val;
    let arr[x] = temp;
    return arr;
}
```



# PART B

## SETUP CLASS

### D R A W X

```
method void draw_x(int loc){
  var int memAddress,location;
  let location = location_fn(loc);
  let memAddress = 16384+location;
  do Memory.poke(memAddress+0, 2);
  do Memory.poke(memAddress+32, 7);
  do Memory.poke(memAddress+64, 14);
  do Memory.poke(memAddress+96, 28);
  do Memory.poke(memAddress+128, 56);
  do Memory.poke(memAddress+160, 112);
  do Memory.poke(memAddress+192, 224);
  do Memory.poke(memAddress+224, 448);
  do Memory.poke(memAddress+256, 896);
  do Memory.poke(memAddress+288, 1792);
  do Memory.poke(memAddress+320, 3584);
  do Memory.poke(memAddress+352, 7168);
  do Memory.poke(memAddress+384, 14336);
  do Memory.poke(memAddress+416, -4096);
  do Memory.poke(memAddress+448, -8192);
  do Memory.poke(memAddress+480, -8192);
  let memAddress = 16384+location+1;
  do Memory.poke(memAddress+0, 16384);
  do Memory.poke(memAddress+32, -8192);
  do Memory.poke(memAddress+64, 28672);
  do Memory.poke(memAddress+96, 14336);
  do Memory.poke(memAddress+128, 7168);
  do Memory.poke(memAddress+160, 3584);
  do Memory.poke(memAddress+192, 1792);
  do Memory.poke(memAddress+224, 896);
  do Memory.poke(memAddress+256, 448);
  do Memory.poke(memAddress+288, 224);
  do Memory.poke(memAddress+320, 112);
  do Memory.poke(memAddress+352, 56);
  do Memory.poke(memAddress+384, 28);
  do Memory.poke(memAddress+416, 15);
  do Memory.poke(memAddress+448, 7);
  do Memory.poke(memAddress+480, 7);
  let memAddress = 16384+location+(32*16);
  do Memory.poke(memAddress+0, -8192);
  do Memory.poke(memAddress+32, -8192);
  do Memory.poke(memAddress+64, -4096);
  do Memory.poke(memAddress+96, 14336);
  do Memory.poke(memAddress+128, 7168);
  do Memory.poke(memAddress+160, 3584);
  do Memory.poke(memAddress+192, 1792);
  do Memory.poke(memAddress+224, 896);
  do Memory.poke(memAddress+256, 448);
  do Memory.poke(memAddress+288, 224);
  do Memory.poke(memAddress+320, 112);
  do Memory.poke(memAddress+352, 56);
  do Memory.poke(memAddress+384, 28);
  do Memory.poke(memAddress+416, 14);
  do Memory.poke(memAddress+448, 7);
  do Memory.poke(memAddress+480, 2);
  let memAddress = 16384+location+(32*16)+1 ;
  do Memory.poke(memAddress+0, 7);
  do Memory.poke(memAddress+32, 7);
  do Memory.poke(memAddress+64, 15);
  do Memory.poke(memAddress+96, 28);
  do Memory.poke(memAddress+128, 56);
  do Memory.poke(memAddress+160, 112);
  do Memory.poke(memAddress+192, 224);
  do Memory.poke(memAddress+224, 448);
  do Memory.poke(memAddress+256, 896);
  do Memory.poke(memAddress+288, 1792);
  do Memory.poke(memAddress+320, 3584);
  do Memory.poke(memAddress+352, 7168);
  do Memory.poke(memAddress+384, 14336);
  do Memory.poke(memAddress+416, 28672);
  do Memory.poke(memAddress+448, -8192);
  do Memory.poke(memAddress+480, 16384);
  return;
}
```



# PART B

## SETUP CLASS

### D R A W O

```
method void draw_o(int loc){
  var int memAddress,location;
  let location = location_fn(loc);
  let memAddress = 16384+location;
  do Memory.poke(memAddress+0, -1024);
  do Memory.poke(memAddress+32, -256);
  do Memory.poke(memAddress+64, -128);
  do Memory.poke(memAddress+96, 8128);
  do Memory.poke(memAddress+128, 2016);
  do Memory.poke(memAddress+160, 496);
  do Memory.poke(memAddress+192, 248);
  do Memory.poke(memAddress+224, 120);
  do Memory.poke(memAddress+256, 60);
  do Memory.poke(memAddress+288, 28);
  do Memory.poke(memAddress+320, 30);
  do Memory.poke(memAddress+352, 14);
  do Memory.poke(memAddress+384, 15);
  do Memory.poke(memAddress+416, 7);
  do Memory.poke(memAddress+448, 7);
  do Memory.poke(memAddress+480, 7);
  let memAddress = 16384+location+1;
  do Memory.poke(memAddress+0, 63);
  do Memory.poke(memAddress+32, 255);
  do Memory.poke(memAddress+64, 511);
  do Memory.poke(memAddress+96, 1016);
  do Memory.poke(memAddress+128, 2016);
  do Memory.poke(memAddress+160, 3968);
  do Memory.poke(memAddress+192, 7936);
  do Memory.poke(memAddress+224, 7680);
  do Memory.poke(memAddress+256, 15360);
  do Memory.poke(memAddress+288, 14336);
  do Memory.poke(memAddress+320, 30720);
  do Memory.poke(memAddress+352, 28672);
  do Memory.poke(memAddress+384, -4096);
  do Memory.poke(memAddress+416, -8192);
  do Memory.poke(memAddress+448, -8192);
  do Memory.poke(memAddress+480, -8192);
  let memAddress = 16384+location+(32*16);
  do Memory.poke(memAddress+0, 7);
  do Memory.poke(memAddress+32, 7);
  do Memory.poke(memAddress+64, 7);
  do Memory.poke(memAddress+96, 15);
  do Memory.poke(memAddress+128, 15);
  do Memory.poke(memAddress+160, 31);
  do Memory.poke(memAddress+192, 30);
  do Memory.poke(memAddress+224, 62);
  do Memory.poke(memAddress+256, 124);
  do Memory.poke(memAddress+288, 248);
  do Memory.poke(memAddress+320, 496);
  do Memory.poke(memAddress+352, 2016);
  do Memory.poke(memAddress+384, 8128);
  do Memory.poke(memAddress+416, -256);
  do Memory.poke(memAddress+448, -1024);
  do Memory.poke(memAddress+480, -4096);
  let memAddress = 16384+location+1+(32*16);
  do Memory.poke(memAddress+0, -8192);
  do Memory.poke(memAddress+32, -8192);
  do Memory.poke(memAddress+64, -8192);
  do Memory.poke(memAddress+96, -4096);
  do Memory.poke(memAddress+128, -4096);
  do Memory.poke(memAddress+160, -2048);
  do Memory.poke(memAddress+192, 30720);
  do Memory.poke(memAddress+224, 31744);
  do Memory.poke(memAddress+256, 15872);
  do Memory.poke(memAddress+288, 7936);
  do Memory.poke(memAddress+320, 3968);
  do Memory.poke(memAddress+352, 2016);
  do Memory.poke(memAddress+384, 1016);
  do Memory.poke(memAddress+416, 255);
  do Memory.poke(memAddress+448, 63);
  do Memory.poke(memAddress+480, 15);
  return;
}
```



# PART B

## SETUP CLASS

This is used to get the location coordinates for each location value given between 1 - 9.

```
method int location_fn(int loc){
    var int location;
    if(loc=1){
        let location = (35*32) + 10;
    }
    if(loc=2){
        let location = (35*32) + 15;
    }
    if(loc=3){
        let location = (35*32) + 19 ;
    }
    if(loc=4){
        let location = (105*32) + 10;
    }
    if(loc=5){
        let location = (105*32) + 15;
    }
    if(loc=6){
        let location = (105*32) + 19;
    }
    if(loc=7){
        let location = (175*32) + 10;
    }
    if(loc=8){
        let location = (175*32) + 15;
    }
    if(loc=9){
        let location = (175*32) + 19;
    }
    return location;
}
```

# PART B

## SETUP CLASS

```
method boolean win(Array arr,int player){
    var int i,j,k;
    let i=0;
    while(~(i=3)){
        if(board(arr,i,0) = player){
            if(board(arr,i,1) = player){
                if(board(arr,i,2) = player){
                    return true;
                }
            }
        }
        let i = i+1;
    }
    let i=0;
    while(~(i=3)){
        if(board(arr,0,i) = player){
            if(board(arr,1,i) = player){
                if(board(arr,2,i) = player){
                    return true;
                }
            }
        }
        let i = i+1;
    }

    if(board(arr,0,0) = player){
        if(board(arr,1,1) = player){
            if(board(arr,2,2) = player){
                return true;
            }
        }
    }
    if(board(arr,0,2) = player){
        if(board(arr,1,1) = player){
            if(board(arr,2,0) = player){
                return true;
            }
        }
    }
    return false;
}
```



# SETUP CLASS

```
method boolean check(Array arr,int player){
    var int i,j,k;
    let i=0;
    while(~(i=3)){
        if(board(arr,i,0) = player){
            if(board(arr,i,1) = player){
                if(board(arr,i,2) = player){
                    return true;
                }
            }
        }
        let i = i+1;
    }
    let i=0;
    while(~(i=3)){
        if(board(arr,0,i) = player){
            if(board(arr,1,i) = player){
                if(board(arr,2,i) = player){
                    return true;
                }
            }
        }
        let i = i+1;
    }
}
```

```
if(board(arr,0,0) = player){
    if(board(arr,1,1) = player){
        if(board(arr,2,2) = player){
            return true;
        }
    }
}
if(board(arr,0,2) = player){
    if(board(arr,1,1) = player){
        if(board(arr,2,0) = player){
            return true;
        }
    }
}
let i=0;
let k = 0;
```

```
while(~(i=3)){
    let j = 0;
    while(~(j=3)){
        if(~(board(arr,i,j) = 0)){
            let k = k+1;
        }
        let j = j+1;
    }
    let i = i+1;
}
if(k=9){
    return true;
}
return false;
```

# PART B

## SETUP CLASS

```
if(computer = 2){
    let temp = 1;
}
else{
    let temp = 2;
}
let win1 = set.win(board1,computer);
let win2 = set.win(board1,temp);
do Output.println(" GAME OVER !!! ");
do Screen.setColor(false);
do Screen.drawRectangle(0,12,511,240);
do Output.moveCursor(11,20);
```

```
    if(win1){
        do Output.println("OOPS!!! You have lost the
game");
        return;
    }
    do Output.moveCursor(11,17);
    if(win2){
        do Output.println("Congragulations!!! You
have won the game");
        return;
    }
    do Output.moveCursor(11,25);
    do Output.println("GAME : DRAW");

    return;
}
```



# PART B

## SETUP CLASS

### GET FUNCTION

```
method int get(){
    var int computer;
    var Setup set;
    let computer = Keyboard.readInt("Do you want to play as x(1) or o(2) : ");
    do set.clear_line();
    if(computer = 1){
        let computer = 2;
    }
    else{
        if(computer = 2){
            let computer = 1;
        }
        else{
            do Output.printString("Please enter a valid number");
            do Sys.wait(2000);
            do set.clear_line();
            let computer = get();
        }
    }
    return computer;
}
```

# PART B

## SETUP CLASS

### MOVE FUNCTION

```
method int move(Array board2,int player,int computer){
    var int temp,i,j,location,k;
    var Setup set;
    do Output.printString(" Thinking...");
    if(computer = 1){
        let temp = 2;
    }
    else{
        let temp = 1;
    }
}
```

```
let i=0;
let j=0;
while(~(i=3)){
    let j=0;
    while(~(j=3)){
        if(set.board(board2,i,j) = 0){
            let board2 =
set.set_board(board2,i,j,computer);
            if(set.win(board2,computer)){
                let location = ((i*3)+j)+1;
                return location;
            }
            let board2 =
set.set_board(board2,i,j,0);
        }
        let j = j+1;
    }
    let i = i+1;
}
```



# PART B

## SETUP CLASS

### MOVE FUNCTION

```
let i=0;
let j=0;
while(~(i=3)){
  let j=0;
  while(~(j=3)){
    if(set.board(board2,i,j) = 0){
      let board2 = set.set_board(board2,i,j,temp);
      if(set.win(board2,temp)){
        let location = ((i*3)+j)+1;
        return location;
      }
      let board2 = set.set_board(board2,i,j,0);
    }
    let j = j+1;
  }
  let i = i+1;
}
```

```
let i = 0;
let j = 0;
let k = 0;
while(~(i=3)){
  let j=0;
  while(~(j=3)){
    if(set.board(board2,i,j) = 0){
      let k = k+1;
    }
    let j = j+1;
  }
  let i = i+1;
}
let location = gekko(board2,k,player,computer);
do set.clear_line();
do Output.moveCursor(0,0);
return location;
```

# PART B

## SETUP CLASS

GEKKO FUNCTION

```
method int gekko(Array board1,int k,int player,int
computer){
    var int location,i,j,k1,pos,max,index;
    var Array values;
    var Setup set;
    let values = Array.new(k);
    let k1 = 0;
    let i=0;
    let j=0;
```

```
while(~(i=3)){
    let j=0;
    while(~(j=3)){
        if(set.board(board1,i,j) = 0){
            let board1 = set.set_board(board1,i,j,player);
            let pos = possibility(board1,player,computer);
            let values[k1] = pos;
            let k1 = k1+1;
            let board1 = set.set_board(board1,i,j,0);
        }
        let j = j+1;
    }
    let i = i+1;
}
```



# PART B

## SETUP CLASS

GEKKO FUNCTION

```
let max = values[0];
let index = 0;
let i = 0;
while(~(i=k)){
    if(max < values[i]){
        let max = values[i];
        let index = i;
    }
    let i = i + 1;
}
let i = 0;
let j=0;
let k1=0;
```

```
while(~(i=3)){
    let j=0;
    while(~(j=3)){
        if(set.board(board1,i,j) = 0){
            if(k1=index){
                let location = ((i*3)+j) + 1;
                return location;
            }
            let k1 = k1+1;
        }
        let j = j+1;
    }
    let i = i+1;
}
return 0;
}
```

# PART B

## SETUP CLASS

### POSSIBILITY FUNCTION

```
method int possibility(Array board1,int player,int
computer){
    var int i,j,k,temp,pos;
    var Setup set;

    let i = 0;
    let j = 0;
    let k = 0;
    while(~(i=3)){
        let j=0;
        while(~(j=3)){
            if(~(set.board(board1,i,j) = 0)){
                let k = k+1;
            }
            let j = j+1;
        }
        let i = i+1;
    }
}
```

```
if(k=9){
    if(computer = 1){
        let temp = 2;
    }
    else{
        let temp = 1;
    }
    if(set.win(board1,computer)){
        return 1;
    }
    else{
        if(set.win(board1,temp)){
            return -1;
        }
        else{
            return 0;
        }
    }
}
```



# PART B

## SETUP CLASS

### POSSIBILITY FUNCTION

```
else{
  if(player = 1){
    let player = 2;
  }
  else{
    let player = 1;
  }
  let pos=0;
  let i = 0;
  let j = 0;
```

```
while(~(i=3)){
  let j=0;
  while(~(j=3)){
    if(set.board(board1,i,j) = 0){
      let board1 = set.set_board(board1,i,j,player);
      let pos = pos + possibility(board1,player,computer);
      let board1 = set.set_board(board1,i,j,0);
    }
    let j = j+1;
  }
  let i = i+1;
}
return pos;
}
```

# PART B

## DOUBLE CLASS

RUN FUNCTION VARIABLE INITIALISATION

```
var Array row1,row2,row3;
var Array board1;
var boolean Game_Over,win1,win2;
var int player,location,row,col,i,j,k;
var Setup set;
let Game_Over = false;
let player = 1;
let row1 = Array.new(3);
let row2 = Array.new(3);
let row3 = Array.new(3);
let board1 = Array.new(3);
```

```
let row1[0] = 0;
let row1[1] = 0;
let row1[2] = 0;
let row2[0] = 0;
let row2[1] = 0;
let row2[2] = 0;
let row3[0] = 0;
let row3[1] = 0;
let row3[2] = 0;
let board1[0] = row1;
let board1[1] = row2;
let board1[2] = row3;
```



# PART B

## DOUBLE CLASS

INPUT LOOP

```
while(~(Game_Over)){  
    if(player = 1){  
        let location = Keyboard.readInt("Player X turn (Enter a number between 1 to 9) : ");  
        do set.clear_line();  
    }  
    else{  
        let location = Keyboard.readInt("Player o turn (Enter a number between 1 to 9) : ");  
        do set.clear_line();  
    }  
}
```

# PART B

## DOUBLE CLASS

### INPUT LOOP

```
if((location<10) & (location>0)){
    let location = location-1;
    let row = location/3;
    let col = location - (3 * (location/3));
    if(set.board(board1,row,col) = 0){
        let location = location+1;
        let board1 = set.set_board(board1,row,col,player);
        if(player = 1){
            do set.draw_x(location);
        }
        else{
            do set.draw_o(location);
        }
    }
}
```

```
let Game_Over = set.check(board1,player);
if(player = 1){
    let player = 2;
}
else{
    let player = 1;
}
}
```



# PART B

## DOUBLE CLASS

INPUT LOOP - EXCEPTION HANDLING

```
else{  
    do Output.println("ERROR : The entered location is already used");  
    do Sys.wait(2000);  
    do set.clear_line();  
}
```

```
else{  
    do Output.println("ERROR : Enter a valid location");  
    do Sys.wait(2000);  
    do set.clear_line();  
}
```

# PART B

## DOUBLE CLASS

### WIN STATEMENT

```
let win1 = set.win(board1,1);
let win2 = set.win(board1,2);
do Output.printString(" GAME OVER !!! ");
do Screen.setColor(false);
do Screen.drawRectangle(0,12,511,240);
do Output.moveCursor(11,20);
if(win1){
    do Output.printString("Player X has won
the game");
    return;
}
```

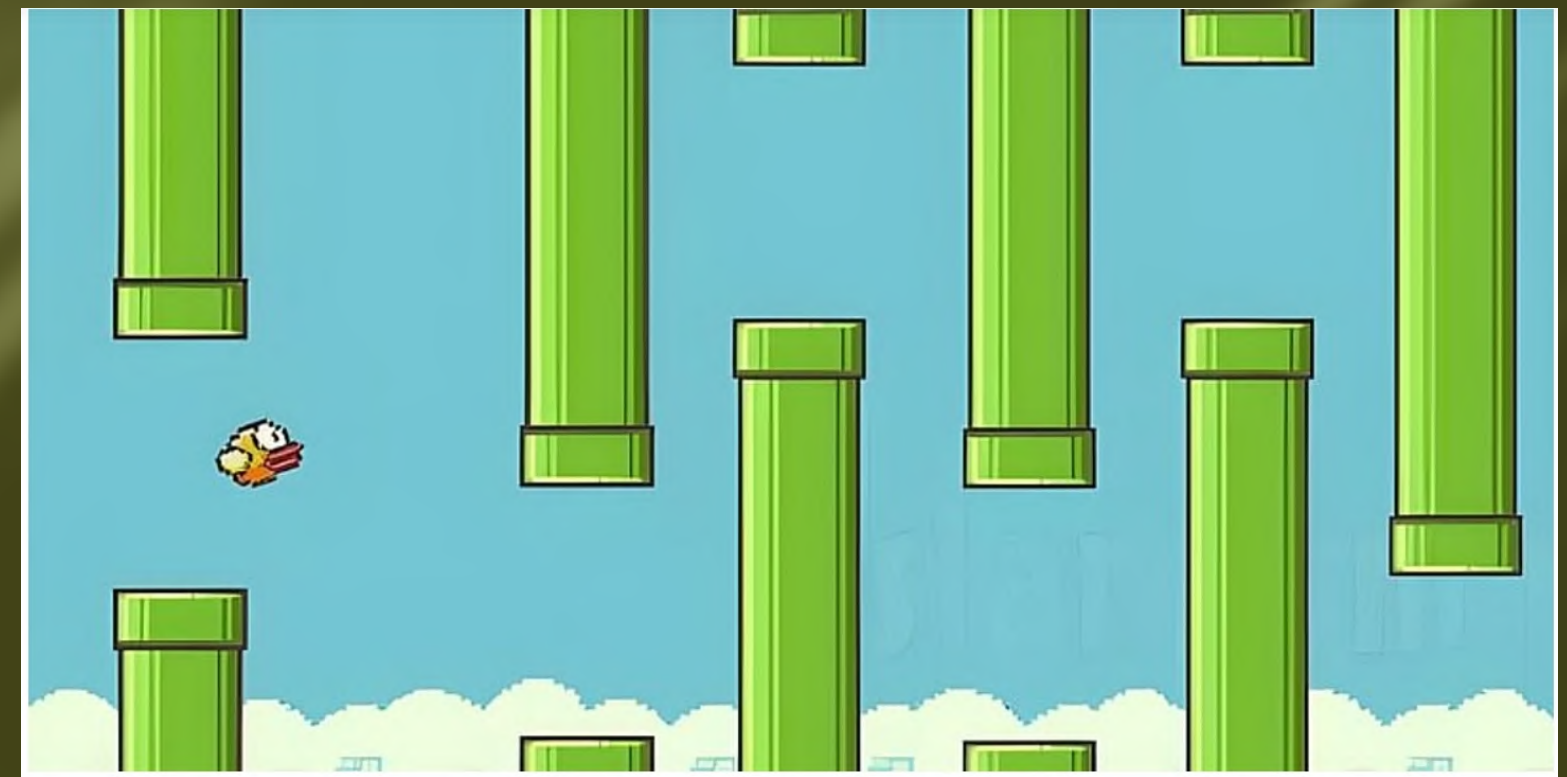
```
if(win2){
    do Output.printString("Player o has won the game");
    return;
}
do Output.moveCursor(11,25);
do Output.printString("GAME : DRAW");
return;
```



# FLAPPY BIRD

# THE CORE IDEA

The objective is to guide the bird safely through the gaps between the pipes without colliding with them. The bird automatically moves forward, and the player's control is limited to tapping a button to make the bird flap its wings and gain upward momentum.



# FLAPPY BIRD

```
class Main {  
    function void main() {  
        var Game game;  
        var boolean run;  
  
        let game = Game.new(2);  
        do game.gamestart();  
        let run = true;  
        while (run) {  
            let run = game.run();  
        }  
        do game.dispose();  
        return;  
    }  
}
```

- The main function is first run creating a variable game of type Game.
- The gamestart method is called without conditions to display the instructions.
- After the loop gets exited, dispose() function is called from the Game class.



# GAME CLASS:

```
field int speed;

field Bird bird;
field Pipe pipe1;
field Pipe pipe2;
field Pipe pipe3;
field Score score;

field char key;
field char last_key;
field boolean first_jump;
field boolean exit;
field boolean reset;
field boolean quit;
field int collision;

field int frame;
```

```
constructor Game new(int s) {
let speed = s;
let player = Player.new(8, 128);
do Pipe.create(2, 50);
let pipe1 = Pipe.new(614+((560/3)*0),
128);
let pipe2 = Pipe.new(614+((560/3)*1),
128);
let pipe3 = Pipe.new(614+((560/3)*2),
128);
let score = Score.new();
return this;
}
```

The instances of types Pipe, Bird are initialized and created.  
And next, the gamestart method is defined which gives instructions to the player before starting.

```

method void gamestart(){
    do Output.moveCursor(0,0);
        do
Output.printString("Use < and >
arrows to increase or decrease
flapping of bird");
        do Output.println();
        do
Output.printString("Press r to
reset");
        do Output.println();
        do
Output.printString("Press q to quit");
        do Output.println();
        do
Output.printString("Press spacebar to
play");
return;
}

```

```

method void inputs() {
let key = Keyboard.keyPressed();
    //
    if (key = 81) {
        let exit = true;
        let quit = true;
    }
    if (key = 82) { // R
        let exit = true;
        let reset = true;
    }
    if ((key = 32) & (~
(last_key=32))) {
. . . . .

```

For each function such as reset, play it's respective ASCII value is given and defined. Then, after checking first jump condition, the instructions are erased.



```
do Output.moveCursor(0,0);
do Output.printString("                                ");
do Output.moveCursor(1,0);
do Output.printString("                                ");
do Output.moveCursor(2,0);
do Output.printString("                                ");
do Output.moveCursor(3,0);
do Output.printString("                                ");
do Output.moveCursor(4,0);
do Output.printString("                                ");
do Output.moveCursor(5,0);
do Output.printString("                                ");
do Output.moveCursor(10, 20);
do Output.printString("                                ");
```

```
method void handle_pipe(int collision) {  
  if (collision =2) {  
    let exit = true;}  
  else{  
    if (collision =1) {  
      do score.increment();  
    }  
  }  
  return;  
}
```

It is checked if the  
bird has hit the  
pipe's boundaries.

```
method void refresh() {  
  do bird.update();  
  if ((~first_jump)) {  
    do pipe1.update();  
    do pipe2.update();  
    do pipe3.update();  
    do score.show();  
  
    do handle_pipe(pipe1.gameover(bird.xP(),bird.yP()));  
    do handle_pipe(pipe2.gameover(bird.xP(),bird.yP()));  
    do handle_pipe(pipe3.gameover(bird.xP(),bird.yP()));
```

The pipes and bird's position are updated and the  
pipe handle function is called.



# GAME CLASS:

```
field int speed;

field Bird bird;
field Pipe pipe1;
field Pipe pipe2;
field Pipe pipe3;
field Score score;

field char key;
field char last_key;
field boolean first_jump;
field boolean exit;
field boolean reset;
field boolean quit;
field int collision;

field int frame;
```

```
constructor Game new(int s) {
let speed = s;
let player = Player.new(8, 128);
do Pipe.create(2, 50);
let pipe1 = Pipe.new(614+((560/3)*0),
128);
let pipe2 = Pipe.new(614+((560/3)*1),
128);
let pipe3 = Pipe.new(614+((560/3)*2),
128);
let score = Score.new();
return this;
}
```

The instances of types Pipe, Bird are initialized and created.  
And next, the gamestart method is defined which gives instructions to the player before starting.



# FLAPPY CLASS:

```
function void draw_digit0(int location) {
  var int memAddress;
  let memAddress = 16384+location;

  do Memory.poke(memAddress+0, -1);
  do Memory.poke(memAddress+32, -1);
  do Memory.poke(memAddress+64, 3);
  do Memory.poke(memAddress+96, 3);
  do Memory.poke(memAddress+128, 3);
  do Memory.poke(memAddress+160, 3);
  do Memory.poke(memAddress+192, 3);
  do Memory.poke(memAddress+224, 3);
  do Memory.poke(memAddress+256, 3);
  do Memory.poke(memAddress+288, 3);
  do Memory.poke(memAddress+320, 15363);
  do Memory.poke(memAddress+352, 15363);
  do Memory.poke(memAddress+384, 15363);
  do Memory.poke(memAddress+416, 15363);
  do Memory.poke(memAddress+448, 15363);
  do Memory.poke(memAddress+480, 15363);
  do Memory.poke(memAddress+512, 15363);
  do Memory.poke(memAddress+544, 15363);
  do Memory.poke(memAddress+576, 15363);
  do Memory.poke(memAddress+608, 15363);
  do Memory.poke(memAddress+640, 15363);
  do Memory.poke(memAddress+672, 15363);
  do Memory.poke(memAddress+704, 15363);
  do Memory.poke(memAddress+736, 15363);
  do Memory.poke(memAddress+768, 15363);
  do Memory.poke(memAddress+800, 15363);
  do Memory.poke(memAddress+832, 3);
  do Memory.poke(memAddress+864, 3);
  do Memory.poke(memAddress+896, 3);
  do Memory.poke(memAddress+928, 3);
  do Memory.poke(memAddress+960, 3);
  do Memory.poke(memAddress+992, 3);
  do Memory.poke(memAddress+1024, 3);
  do Memory.poke(memAddress+1056, 3);
  do Memory.poke(memAddress+1088, -1);
```

```
do Memory.poke(memAddress+1, 255);
do Memory.poke(memAddress+33, 255);
do Memory.poke(memAddress+65, 192);
do Memory.poke(memAddress+97, 192);
do Memory.poke(memAddress+129, 192);
do Memory.poke(memAddress+161, 192);
do Memory.poke(memAddress+193, 192);
do Memory.poke(memAddress+225, 192);
do Memory.poke(memAddress+257, 192);
do Memory.poke(memAddress+289, 192);
do Memory.poke(memAddress+321, 192);
do Memory.poke(memAddress+353, 192);
do Memory.poke(memAddress+385, 192);
do Memory.poke(memAddress+417, 192);
do Memory.poke(memAddress+449, 192);
do Memory.poke(memAddress+481, 192);
do Memory.poke(memAddress+513, 192);
do Memory.poke(memAddress+545, 192);
do Memory.poke(memAddress+577, 192);
do Memory.poke(memAddress+609, 192);
do Memory.poke(memAddress+641, 192);
do Memory.poke(memAddress+673, 192);
do Memory.poke(memAddress+705, 192);
do Memory.poke(memAddress+737, 192);
do Memory.poke(memAddress+769, 192);
do Memory.poke(memAddress+801, 192);
do Memory.poke(memAddress+833, 192);
do Memory.poke(memAddress+865, 192);
do Memory.poke(memAddress+897, 192);
do Memory.poke(memAddress+929, 192);
do Memory.poke(memAddress+961, 192);
do Memory.poke(memAddress+993, 192);
do Memory.poke(memAddress+1025, 192);
do Memory.poke(memAddress+1057, 192);
do Memory.poke(memAddress+1089, 255);
do Memory.poke(memAddress+1121, 255);
return;
```



# FLAPPY CLASS:

Bitmap

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1																
2																
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
16																

Rotate right

Vertical Mirror

Function Type:

function ▼

Function Name:

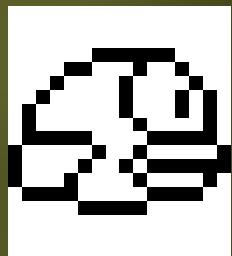
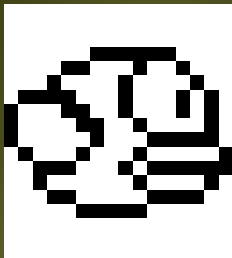
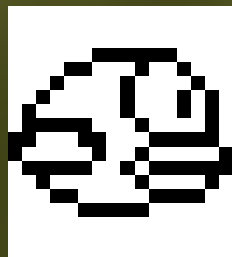
draw

Generate Code >>

Generated Jack Code

```
function void draw(int location) {  
    let memAddress = 16384+location;  
    do Memory.poke(memAddress+0, 0);  
    do Memory.poke(memAddress+32, 4032);  
    do Memory.poke(memAddress+64, 4656);  
    do Memory.poke(memAddress+96, 26632);  
    do Memory.poke(memAddress+128, 19486);  
    do Memory.poke(memAddress+160, 16417);  
    do Memory.poke(memAddress+192, -959);  
    do Memory.poke(memAddress+224, -32191);  
    do Memory.poke(memAddress+256, 31794);  
    do Memory.poke(memAddress+288, 24652);  
    do Memory.poke(memAddress+320, 8064);  
    do Memory.poke(memAddress+352, 0);  
    do Memory.poke(memAddress+384, 0);  
    do Memory.poke(memAddress+416, 0);  
    do Memory.poke(memAddress+448, 0);  
    do Memory.poke(memAddress+480, 0);  
    return;  
}
```

# FLAPPY CLASS:



9 8 7 6 5 4 3 2 1 0



# BIRD CLASS:

```
class Bird{  
  field int row;  
  field int col;  
  field int velocity;  
  field int index;  
  field int fly;  
  field int time;  
  field int up;  
  field int down;
```

```
  constructor Bird new(int xP, int yP) {  
    do setbird(xP, yP);  
    return this;  
  }  
  method int xP() {  
    return col;  
  }  
  method int yP() {  
    return row;  
  }  
}
```

# BIRD CLASS:

```
class Bird{  
  field int row;  
  field int col;  
  field int velocity;  
  field int index;  
  field int fly;  
  field int time;  
  field int up;  
  field int down;
```

```
  constructor Bird new(int xP, int yP) {  
    do setbird(xP, yP);  
    return this;  
  }  
  method int xP() {  
    return col;  
  }  
  method int yP() {  
    return row;  
  }
```



# BIRD CLASS:

```
method void setbird(int xP, int yP) {  
  let col = 128;  
  let row = yP;  
  let velocity=0;  
  let index = ((32*row)+ (col/16));  
  let fly = 0;  
  let time = 0;  
  let down = 1;  
  let up = 0;  
  return;  
}
```

```
method void jump() {  
  let velocity = -4;  
  if(down = 1){  
    let down = 0; //true  
  }  
  return;  
}
```

# BIRD CLASS:

```
method void updbird(){
```

```
let time=time+1;
```

```
if (time = 5) {
```

```
    let time = 0;
```

```
    let fly = fly + 1;
```

```
        if (velocity < 6){
```

```
            let velocity=velocity+1;
```

```
        }
```

```
        if (fly = 4){
```

```
            let fly = 0;
```

```
    }
```

```
if (down = 0){
```

```
    if( velocity > 0) {
```

```
        if(row < (221-velocity)){
```

```
let row = row +velocity;
```

```
let index = index + (velocity*32);
```

```
    }
```

```
    }
```

```
else {
```

```
if (row > -velocity){
```

```
let row=row+velocity;
```

```
let index= index+ (velocity*32);
```

```
    } else {
```

```
let velocity = 0;
```

```
    }
```

```
    }
```



```
    else {
        if (up = 0) {
            if ((time = 1)) {
                let row = row - 1;
                let index = index - 32;
            }
            if (row < 118)
            {let up = 1;
            }
        }

        } else {
            if ((time = 1)) {
                let row = row + 1;
                let index = index + 32;
                if (row > 138) {let up = 0;}}
            }
        }

    }
    return;
}
```

# BIRD CLASS:

```
method void render() {  
    if (fly = 0) {  
        do Flappy.flappy1(index); }  
        else {  
if (fly = 1) {  
do Flappy.flappy2(index); }  
        else {  
    if (fly = 2) {  
do Flappy.flappy3(index); }  
        else {  
    if (fly = 3) {  
do Flappy.flappy2(index); }  
        } } }  
    return;  
}
```

```
method void update() {  
    do updbird();  
    do render();  
    return;  
}
```



# MOD CLASS:

```
class Mod {  
  
    function int mod(int a, int b) {  
        var int qoutient;  
        var int reminder;  
        let qoutient = (a / b);  
        let reminder = a - (b * qoutient);  
        return reminder;  
    }  
}
```

# PIPE CLASS:

```
static int xvelocity;
static int fixedgap;

field int xposition;
field int yposition;
field int time; // internal clock
field boolean pass; // passed Bird?

function void create(int x, int y) {
    let xvelocity = x;
    let fixedgap = y;
    return;
}

constructor Pipe new(int x, int y) {

    let xposition = x;
    let yposition = y;
    let time = -1;
    let pass = true;
    return this;
```



# PIPE CLASS:

```
do Screen.drawRectangle(Math.min(Math.max(xposition-2,0),511), yposition+30,  
Math.min(Math.max(xposition+44,0),511), yposition+31); // top side  
do Screen.drawRectangle(Math.min(Math.max(xposition-2,0),511), yposition+54,  
Math.min(Math.max(xposition+44,0),511), yposition+55); // bottom side  
do Screen.drawRectangle(Math.min(Math.max(xposition-2,0),511), yposition+30,  
Math.min(Math.max(xposition-1,0),511), yposition+54); // left side  
do Screen.drawRectangle(Math.min(Math.max(xposition+44,0),511), yposition+30,  
Math.min(Math.max(xposition+45,0),511), yposition+55); // right side
```

# PIPE CLASS:

```
do Screen.drawRectangle(Math.min(Math.max(xposition,0),511), yposition+54,  
Math.min(Math.max(xposition+1,0),511), 255);           // left wall  
do Screen.drawRectangle(Math.min(Math.max(xposition+42,0),511), yposition+54,  
Math.min(Math.max(xposition+43,0),511), 255);          // right wall
```



# PIPE CLASS:

```
do Screen.drawRectangle(Math.min(Math.max(xposition-2,0),511), yposition+(-fixedgap),  
Math.min(Math.max(xposition+45,0),511), yposition+(-(fixedgap-1)));      // top  
    do Screen.drawRectangle(Math.min(Math.max(xposition-2,0),511), yposition+(-(fixedgap+25)),  
Math.min(Math.max(xposition+45,0),511), yposition+(-(fixedgap+24))); // bottom  
    do Screen.drawRectangle(Math.min(Math.max(xposition-2,0),511), yposition+(-(fixedgap+23)),  
Math.min(Math.max(xposition-1,0),511), yposition+(-fixedgap));          // left  
    do Screen.drawRectangle(Math.min(Math.max(xposition+44,0),511), yposition+(-(fixedgap+25)),  
Math.min(Math.max(xposition+45,0),511), yposition+(-fixedgap));        // right
```

# PIPE CLASS:

```
do Screen.drawRectangle(Math.min(Math.max(xposition,0),511), 0,  
Math.min(Math.max(xposition+1,0),511), yposition+(-(fixedgap+25))); // left  
do Screen.drawRectangle(Math.min(Math.max(xposition+42,0),511), 0,  
Math.min(Math.max(xposition+43,0),511), yposition+(-(fixedgap+25))); // right
```



# PIPE CLASS:

```
do Screen.setColor(false);

    do Screen.drawRectangle(Math.min(Math.max(xposition,0),511), yposition+32,
Math.min(Math.max(xposition+(1+xvelocity),0),511), yposition+53);
    do Screen.drawRectangle(Math.min(Math.max(xposition+46,0),511), yposition+30,
Math.min(Math.max(xposition+(48+xvelocity),0),511), yposition+55);
    do Screen.drawRectangle(Math.min(Math.max(xposition+2,0),511), yposition+56,
Math.min(Math.max(xposition+(4+xvelocity),0),511), 255);
    do Screen.drawRectangle(Math.min(Math.max(xposition+44,0),511), yposition+56,
Math.min(Math.max(xposition+(46+xvelocity),0),511), 255);
```

# PIPE CLASS:

```
method void updpipe() {  
    let time = time + 1;  
  
    if (time = 5) {  
        let time = 0;  
    } else {  
        let xposition = xposition - xvelocity;  
  
        if (xposition < (-50)) {  
            let pass = true;  
            let xposition = 514;  
            let yposition = LCGRandom.randrange(80,194);  
        }  
  
        if (xposition > 614) {  
            let yposition = LCGRandom.randrange(80,194);  
        }  
    }  
    return;  
}
```



# PIPE CLASS:

```
method void update() {  
    do updpipe();  
    do draw();  
}
```

```
method int x() {  
    return xposition;  
}
```

```
method int y() {  
    return yposition;  
}
```

```
method int gameover(int x, int y) {  
    if ((x>(xposition-32))&(x<(xposition+40))) {  
        if ((y>yposition)|(y<(yposition-(fixedgap+6)))) {  
            return 2;  
        }  
        if (pass) {  
            let pass = false;  
            return 1;  
        }  
        else {return 0;}  
    }  
    return 0;  
}
```

# PIPE CLASS:

```
method void updpipes() {  
    let time = time + 1;  
  
    if (time = 5) {  
        let time = 0;  
    } else {  
        let xposition = xposition - xvelocity;  
  
        if (xposition < (-50)) {  
            let pass = true;  
            let xposition = 514;  
            let yposition = LCGRandom.randrange(80,194);  
        }  
  
        if (xposition > 614) {  
            let yposition = LCGRandom.randrange(80,194);  
        }  
    }  
    return;  
}
```



# SCORE CLASS:

```
field int total;
field int v1;
field int v2;
field int v3;

constructor Score new() {
    do reset();
    return this;
}

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

```
method void render(int x, int v) {
    if (v = 0) {do Flappy.draw_digit0(x+512);} else {
    if (v = 1) {do Flappy.draw_digit1(x+512);} else {
    if (v = 2) {do Flappy.draw_digit2(x+512);} else {
    if (v = 3) {do Flappy.draw_digit3(x+512);} else {
    if (v = 4) {do Flappy.draw_digit4(x+512);} else {
    if (v = 5) {do Flappy.draw_digit5(x+512);} else {
    if (v = 6) {do Flappy.draw_digit6(x+512);} else {
    if (v = 7) {do Flappy.draw_digit7(x+512);} else {
    if (v = 8) {do Flappy.draw_digit8(x+512);} else {
    if (v = 9) {do Flappy.draw_digit9(x+512);} else {
    }}}}}}}}}}}
    return;
}
```

# SCORE CLASS:

```
method void show() {
    if (total > 99) {
        do render(16,v1);
        do render(14,v2);
        do render(12,v3);
    } else {
        if (total > 9) {
            do render(16,v1);
            do render(14,v2);
        }
        else {
            if (total > 0) {do render(15,v1);}
        }
    }
    return;
}
```

```
method void show() {
    if (total > 99) {
        do render(16,v1);
        do render(14,v2);
        do render(12,v3);
    } else {
        if (total > 9) {
            do render(16,v1);
            do render(14,v2);
        }
        else {
            if (total > 0) {do render(15,v1);}
        }
    }
    return;
}
```



# SCORE CLASS:

```
method void reset() {  
    let v1 = 0;  
    let v2 = 0;  
    let v3 = 0;  
    let total = 0;  
    return;  
}
```

# SCORE CLASS:

```
method void reset() {  
    let v1 = 0;  
    let v2 = 0;  
    let v3 = 0;  
    let total = 0;  
    return;  
}
```



# LCGRANDOM CLASS:

The `setSeed` function is used to set the seed value for the random number generator. It takes an integer parameter `newSeed` and assigns it to the seed variable. If the seed value is 0, it is set to 1 to avoid potential issues with the LCG algorithm. Additionally, it calculates the values of A, M, Q, and R based on fixed formulas. The return statement indicates that the function does not return any value.

# LCGRANDOM CLASS:

The `rand` function generates a random integer between 0 and  $M-1$  (inclusive) using the LCG algorithm. It calculates the intermediate value `test` by applying the LCG formula, which involves multiplying `A` by the modulo of `seed` and `Q`, and subtracting `R` multiplied by the integer division of `seed` by `Q`. If `test` is negative, it adds `M` to it to ensure the result is in the desired range. Finally, it updates the `seed` value and returns it.



# LCGRANDOM CLASS:

The `rand` function generates a random integer between 0 and  $M-1$  (inclusive) using the LCG algorithm. It calculates the intermediate value `test` by applying the LCG formula, which involves multiplying `A` by the modulo of `seed` and `Q`, and subtracting `R` multiplied by the integer division of `seed` by `Q`. If `test` is negative, it adds `M` to it to ensure the result is in the desired range. Finally, it updates the `seed` value and returns it.

# LCGRANDOM CLASS:

The `randRange` function generates a random integer within a given range, defined by the parameters `low` and `high`. It first calculates the scale variable, which represents the scaling factor needed to map the random number range to the desired range. It then calls the `rand` function to generate a random number between 0 and `M-1` and divides it by `scale`. This value is then added to `low` to obtain a random number within the specified range.

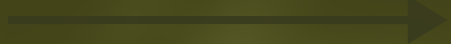


ROCK PAPER AND SCISSOR  
GAME

READY

# ABOUT THE GAME

it is a double player game where each player has three choices. the choices for the player1 and player2 is 0-Rock,1-Paper and 2-scissor. a simple way to explain the logic is the value of rock(0) is lesser than the value of paper(1).the value of paper(1) is lesser than the value of scissor(2). the value of scissor(2) is lesser than the value of rock(0). if user enters same numbers for both players then it is a tie match. winners of the match will be based on the above logic. if the user enters wrong input then the output is invalid.





in this the codes are divided into three classes

- 1) main class
- 2) GUI\_R class
- 3) GUI\_L class

1) main class :

in this we first declare variables variables such as player 1,player 2,sets, pl1score,pl2score,tiescore,a.Initially the scores of each player is zero and uses a loop to ask the user for the number of sets they want to play. The code uses various function calls like "Output.moveCursor", "Output.printString", " Keyboard.readInt", " Keyboard.readInt", "Screen.clearScreen"," GUI\_L.draw\_L", and "GUI\_R.draw\_R". These functions are likely part of a custom library used for input/output operations and graphical user interface.

# DATE



# CODE IS

```
class Main {

    function void main(){
    var int player1;
    var int player2;
    var int sets;
    var int pl1score;
    var int pl2score;
    var int tiescore;
    var int a;

    let pl1score = 0;
    let pl2score = 0;
    let tiescore = 0;
    do Output.moveCursor(3,12);
    do Output.printString("ROCK-PAPER-SCISSOR");
    do Output.moveCursor(6,22);
    let sets = Keyboard.readInt("How many sets: ");

    let a=1;
    while((sets > a) | (sets = a)){

        do Screen.clearScreen();
        do Output.moveCursor(0,0);

        let a = a+1;
        do Output.moveCursor(0, 15);
        do Output.printString("-- PLAYERS CHOICES -");
        do Output.println();
        do Output.moveCursor(3, 12);
        do Output.printString(" 0 - ROCK || 1 - PAPER || 2- SCISSOR");
        do Output.println();
        do Output.moveCursor(5, 0);
        do Output.printString(" GAME SET: ");
        do Output.printInt(a - 1);
        do Output.println();
        do Output.println();
    }
}
```

Try Pitch

```
do Output.moveCursor(7, 22);
let player1 = Keyboard.readInt("PLAYER 1 PICK : ");

do Screen.clearScreen();
do Output.moveCursor(0, 15);
do Output.printString("-- PLAYERS CHOICES ---");
do Output.println();
do Output.moveCursor(3, 12);
do Output.printString("0 - ROCK || 1 - PAPER || 2 - SCISSOR");
do Output.println();
do Output.moveCursor(5, 0);
do Output.printString(" GAME SET: ");
do Output.printInt(a - 1);
do Output.println();
do Output.println();
do Output.moveCursor(7, 22);
let player2 = Keyboard.readInt("PLAYER 2 PICK: ");

do Screen.clearScreen();
do Output.moveCursor(6, 0);
do GUI_L.draw_L(player1);
do Output.moveCursor(6, 32);
do GUI_R.draw_R(player2);
do Output.println();
do Output.moveCursor(18, 20);
if((player1 = 0)&(player2 = 0)){
    do Output.printString("it's a tie");
    let tiescore = tiescore + 1;
}
if((player1 = 1)&(player2 = 1)){
    do Output.printString("it's a tie");
    let tiescore = tiescore + 1;
}
if((player1 = 2)&(player2 = 2)){
    do Output.printString("it's a tie");
    let tiescore = tiescore + 1;
}
```

# DATE



```

    let pl2score = pl2score + 1;
}

if((player1 = 0)&(player2 = 1)){
    do Output.printString(" player2 wins");
    let pl2score = pl2score + 1;
}

if((player1 = 1)& (player2 = 2)){
    do Output.printString(" player2 wins");
    let pl2score = pl2score + 1;
}

if((player1 = 2)&(player2 = 0)){
    do Output.printString(" player2 wins");
    let pl2score = pl2score + 1;
}

if((player1 = 1)&(player2 = 0)){
    do Output.printString(" player1 wins");
    let pl1score = pl1score + 1;
}

if((player1 = 2)&(player2 = 1)) {
    do Output.printString(" player1 wins");
    let pl1score = pl1score+ 1;
}

if((player1 = 0)&(player2 = 2)){
    do Output.printString("player1 wins");
    let pl1score = pl1score + 1;
}

if(player1>2){
    do Output.printString("invalid");
}

if(player2>2){
    do Output.printString("invalid");
}

do Sys.wait(5000);

```

```

}
do Screen.clearScreen();
if(pl1score > pl2score){
    do Output.println();
    do Output.moveCursor(8, 15);
    do Output.printString("<----- -Score Summary ----- >");
    do Output.println();
    do Output.println();
    do Output.moveCursor(11, 15);
    do Output.printString(" player1 : ");
    do Output.printInt(pl1score);
    do Output.println();
    do Output.println();
    do Output.moveCursor(14, 15); do Output.printString(" player2 : ");
    do Output.printInt(pl2score);
    do Output.println();
    do Output.println();
    do Output.moveCursor(17, 15);
}

if(pl2score > pl1score) {
    do Output.println();
    do Output.moveCursor(8, 15);
    do Output.printString(" <--- scroe summary---> ");
    do Output.println();
    do Output.println();
    do Output.moveCursor(11, 15);
    do Output.printString(" player1 : ");
    do Output.printInt(pl1score);
    do Output.println();
    do Output.println();
    do Output.moveCursor(14, 15);
    do Output.printString(" player2 : ");
    do Output.printInt(pl2score);
    do Output.println();
    do Output.println();
    do Output.moveCursor(17, 15);
}

```



first the user enters the input. after After receiving the input from both players, the code clears the screen and displays the output in a diagram from. For clearing the screen we use a function call "Screen.clearScreen". Here we used a function called "sys.wait" to give some time for outputs before starting the next set of game. we use key words like do let.. After completing the sets the code clears the screen again and displays. mostly the is code by using if and while conditions.and

D

```
if(pl1score = pl2score) {  
    do Output.println();  
    do Output.moveCursor(8, 15);  
    do Output.printString("<---Score Summary--->");  
    do Output.println();  
    do Output.println();  
    do Output.moveCursor(11, 15);  
    do Output.printString(" player1 : ");  
    do Output.printInt(pl1score);  
    do Output.println();  
    do Output.println();  
    do Output.moveCursor(14, 15);  
    do Output.printString(" player2 : ");  
    do Output.printInt(pl2score );  
    do Output.println();  
    do Output.println();  
    do Output.moveCursor(17, 15);  
    do Output.printString(" Overall it is a tie.");  
}  
return;
```



GUI\_R class:

This code defines a class called GUI\_R which contains a function called draw\_R. The purpose of this function is to draw a visual representation of a game character. The function draw\_R takes an integer called player. there are multiple if statements that check the value of the player who is giving the input. the diagrams will be displayed in the right side of the screen.

If player is equal to 0, it means the character is "ROCK".an if condition is there to check. The then code uses the Output object to print a visual representation of the character.

If player is equal to 1, it means the character is "PAPER". an if condition is there to check. The code then uses the Output object to print a visual representation of the character.

If player is equal to 2, it means the character is "SCISSOR".an if condition is there to check. The then code uses the Output object to print a visual representation of the character.

If the user enters an invalid number then the output will be shown as invalid. no character will be displayed



```

class GUI_R {
    function void draw_R(int player){
        if (player = 0){
            do Output.printString(" ROCK");
            do Output.println();
            do Output.moveCursor(7, 32);
            do Output.printString("      _____");
            do Output.println();
            do Output.moveCursor(8, 32);
            do Output.printString(" __(| ");
            do Output.println();
            do Output.moveCursor(9, 32);
            do Output.printString("(____) ");
            do Output.println();
            do Output.moveCursor(10, 32);
            do Output.printString("(____) ");
            do Output.println();
            do Output.moveCursor(11, 32);
            do Output.printString("(____) ");
            do Output.println();
            do Output.moveCursor(12, 32);
            do Output.printString("(____)_____");
            do Output.println();
        }

        if (player = 1){
            do Output.printString(" PAPER");
            do Output.println();
            do Output.moveCursor(7, 32);
            do Output.printString("      _____");
            do Output.println();
            do Output.moveCursor(8, 32);
            do Output.printString(" __(|____| ");
            do Output.println();
            do Output.moveCursor(9, 32);
            do Output.printString(" __(|____) ");
            do Output.println();
            do Output.moveCursor(10, 32);
            do Output.printString("(____) ");

```

```

            do Output.printString("(____) ");
            do Output.println();
            do Output.moveCursor(11, 32);
            do Output.printString(" (____) ");
            do Output.println();
            do Output.moveCursor(12, 32);
            do Output.printString(" (____)_____");
            do Output.println();
        }

        if (player = 2) {
            do Output.printString(" SCISSOR");
            do Output.println();
            do Output.moveCursor(7, 32);
            do Output.printString("      _____");
            do Output.moveCursor(8, 32);
            do Output.printString(" _____(|____");
            do Output.println();
            do Output.moveCursor(9, 32);
            do Output.printString(" (____) ");
            do Output.println();
            do Output.moveCursor(10, 32);
            do Output.printString(" (____)");
            do Output.println();
            do Output.moveCursor(11, 32);
            do Output.printString(" (____)");
            do Output.println();
            do Output.moveCursor(12, 32);
            do Output.printString(" (____)_____");
            do Output.println();
        }

        return ;
    }
}

```



# `_L CLASS :`

This code defines a class called GUI\_L which contains a function called draw\_L. The purpose of this function is to draw a visual representation of a game character. The function draw\_L takes an integer called player. there are multiple if statements that check the value of the player who is giving the input. the diagrams will be displayed in left side of the screen

If player is equal to 0, it means the character is "ROCK".an if condition is there to check. The then code uses the Output object to print a visual representation of the character.

If player is equal to 1, it means the character is "PAPER". an if condition is there to check. The code then uses the Output object to print a visual representation of the character.

If player is equal to 2, it means the character is "SCISSOR".an if condition is there to check. The then code uses the Output object to print a visual representation of the character.

If the user enters an invalid number then the output will be shown as invalid. no character will be displayed

```

class GUI_L {
function void draw_L(int player){
    if (player = 0) {

        do Output.printString(" ROCK");
        do Output.println();
        do Output.printString("    ");
        do Output.println();
        do Output.printString("    |__)|_");
        do Output.println();
        do Output.printString("    (____)");
        do Output.println();
        do Output.printString("    (____)");
        do Output.println();
        do Output.printString("    (____)");
        do Output.println();
        do Output.printString("    _____(____)");
        do Output.println();
    }
    if (player = 1){

        do Output.printString(" PAPER");
        do Output.println();
        do Output.printString("    ");
        do Output.println();
        do Output.printString("__|    )__");
        do Output.println();
        do Output.printString("    (____)");
        do Output.println();
        do Output.printString("    (____)");
        do Output.println();
        do Output.printString("    (____)");
        do Output.println();
        do Output.printString("_____,____");
        do Output.println();
    }
}

```

```

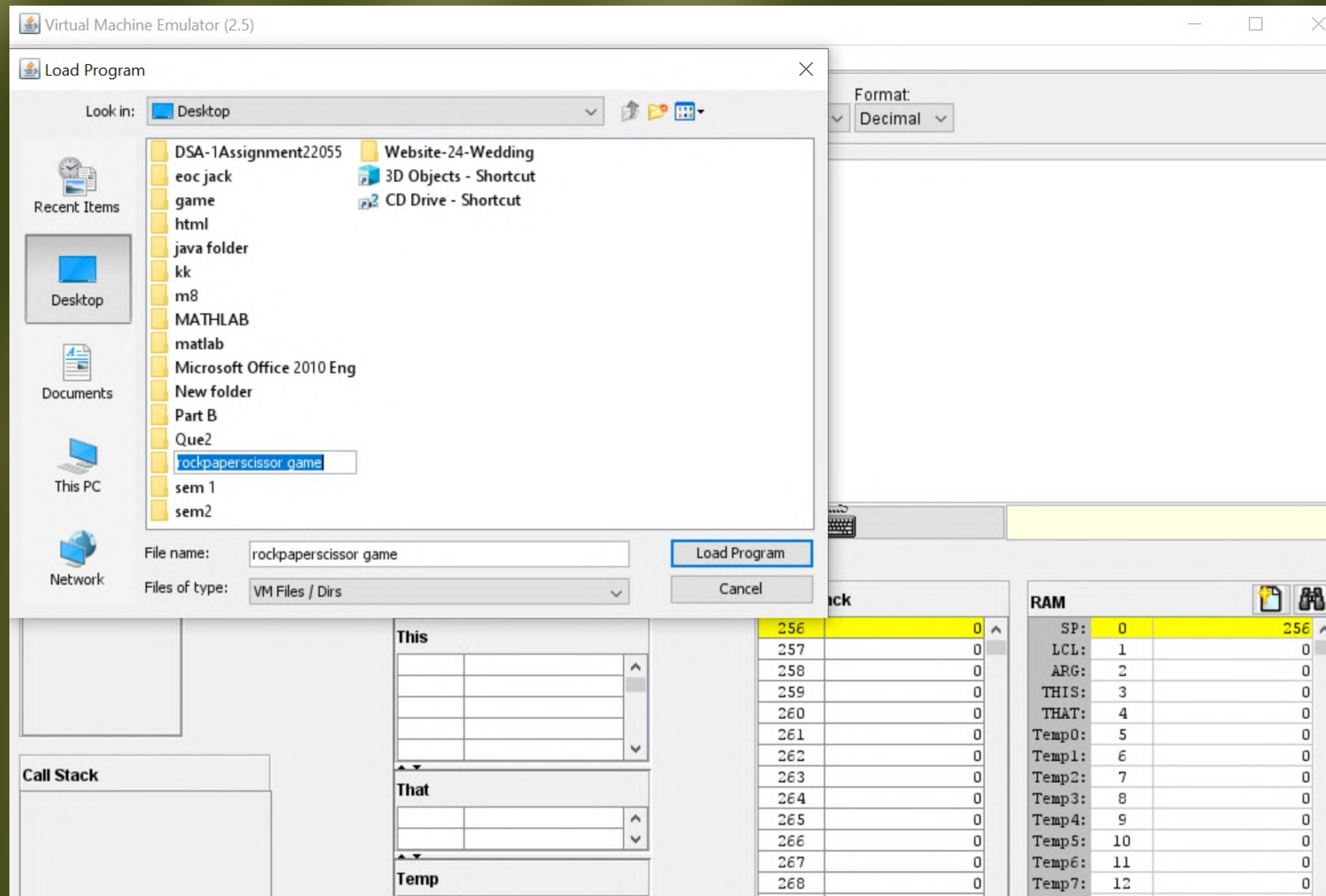
    if (player = 2){
        do Output.printString(" SCISSOR");
        do Output.println();
        do Output.printString("    ");
        do Output.println();
        do Output.printString("__|    )__");
        do Output.println();
        do Output.printString("    (____)");
        do Output.println();
        do Output.printString("    (____)");
        do Output.println();
        do Output.printString("    (____)");
        do Output.println();
        do Output.printString("_____(____)");
        do Output.println();
    }

    return;
}
}

```



# VAILD OUTPUTS



```
-- PLAYERS CHOICES --  
  
0 - ROCK || 1 - PAPER || 2- SCISSOR  
  
GAME SET: 1  
  
PLAYER 1 PICK : 1■
```

```
-- PLAYERS CHOICES ---  
  
0 - ROCK || 1 - PAPER || 2 - SCISSOR  
  
GAME SET: 1  
  
PLAYER 2 PICK: 1■
```

PAPER

```
  |-----)
-----|
  {-----)
  {-----)
  {-----)
-----|
```

PAPER

```
  (-----|
  (-----)
  (-----)
  (-----)
  (-----)
-----|
```

it's a tie

<---Score Summary--->

player1 : 0

player2 : 0

Overall it is a tie.

here the user enters same value for both players. so that ts a tie no one wins the match. and the diagram is displayed there. score for each player is displayed.



# T N V A L I D   C A S E

-- PLAYERS CHOICES -

0 - ROCK || 1 - PAPER || 2- SCISSOR

GAME SET: 1

PLAYER 1 PICK : 4■

-- PLAYERS CHOICES ---

0 - ROCK || 1 - PAPER || 2 - SCISSOR

GAME SET: 1

PLAYER 2 PICK: 0■

ROCK

(-----  
(-----  
(-----  
(-----  
(-----  
(-----

invalid

THANK YOU