Project breakdown

- **Encoding the maze and robot position:**
  - Store the maze in MIPS (what data structure should we use, and how to record wall and path in the data, given that MIPS does not have complicated default data structure such as list, dictionary)
  - Record the robot position in MIPS (Maybe need to use several variables to record its coordinate)
- **Taking input and move the robot:**
  - Provide instructions for user input
  - Transfer user input like “F”, “B”, “L”, “R” into MIPS order and change relevant variables accordingly
- **Feasibility check:**
  - Check a movement is valid or not
  - Provide feedback as a print output
- **Exit check:**
  - Once the robot reaches destination, need to know and stop the program
- **Step and mistake count:**
  - Once reaching destination, print out the number of steps taken and mistakes made.

Current progress 1: James completed a demo project with same functions in Python

- Encoding the maze by recording all feasible directions in each cell

```
maze = [
    [[], [], [], [], [], []],
    [[], ["F"], ["B","R"], ["F"], ["B", "R", "F"], ["B", "L"]],
    [[], ["R", "F"], ["B", "L", "R", "F"], ["B"], ["L", "R"], ["R"]],
    [[], ["L"], ["L", "F"], ["B", "R", "F"], ["B", "L"], ["L"]],
    [[], ["R", "F"], ["B", "R", "F"], ["B", "L", "R"], ["F"], ["B", "L", "R"]],
    [[], ["L"], ["L", "R"], ["L", "F"], ["B", "R", "F"], ["B", "L"]],
    [["F"], ["F"], ["B", "L", "F"], ["B"], ["L", "F"], ["B"]]
]
```

- Taking input and move the robot
- Feasibility check
- Exit check
- Step and mistake count

```
print("Enter a direction: R for right, L for left, f for forward, and b for backwards: ")

while (x_position, y_position) != (0,5):
    user_input = input("")
    if user_input in maze[x_position][y_position]:
        x_position += moves_x[user_input]
        y_position += moves_y[user_input]
        count += 1
    else:
        print("Invalid move! Try again...")
        count += 1
        mistakes += 1

print("Congratulation! You reached the exit.")
print("Number of mistakes: ", mistakes)
print("Total number of moves: ", count)
```

Current progress 2: One possible solution to encode the maze in MIPS

- Following the python project, we tried to encode the maze by encoding 4 directions of each cell.
- The 2-D maze is encoded in a list of 1-D 4digits numbers, and use y and x coordinates to locate the number
- In each cell the 4digits number represent “F, R, B, L” respectively, where “1” stands for a feasible path and ”9” stands for a wall

First row is left empty First colume is left empty [1,1] [1,2] [1,3] [1,4]

```
.data
myarray: .word 9999,9999,9999,9999,9999,9999,9999,1999,9119,1999,1119,9911,9999,1199,
# 9 stands for wall, 1 stands for path
mazewidth: .word 6
ycord: .word 1 #can be changed
xcord: .word 2 #can be changed


.text
lw $s0, xcord
lw $s1, ycord
lw $s2, mazewidth

mul $t0,$s1,$s2
add $t1,$t0,$s0

sll $t1,$t1,2

la $t2, myarray
add $t3,$t2,$t1

lw $a0, 0($t3)
li $v0, 1
syscall
```

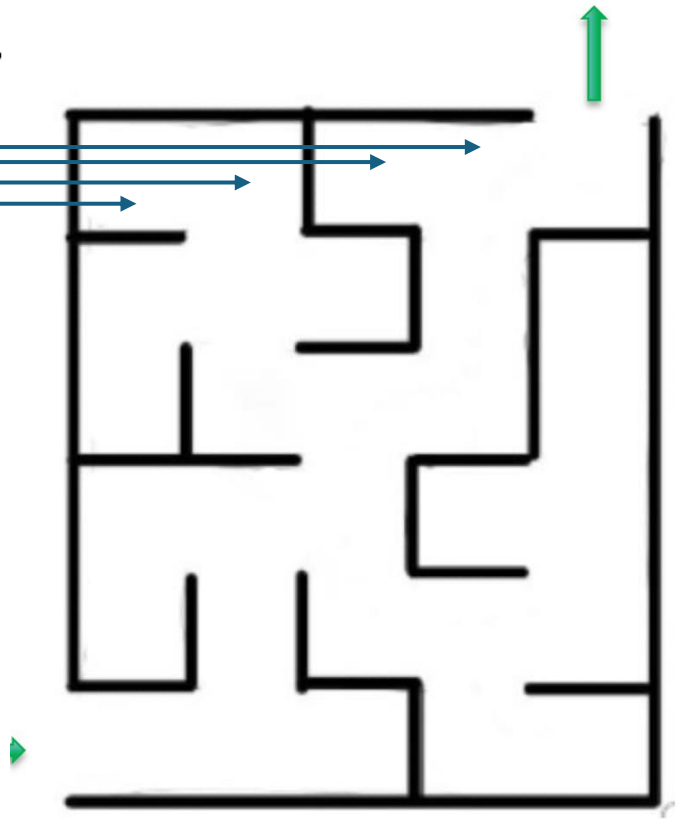

Figure 1: Maze map

Potential To-Dos

- Other parts of the task remaining unfinished in MIPS:
  - Taking user input and move the robot
  - Feasibility check for each step
  - Step and mistake count
  - Exit check and print result
- There might be some other ways of encoding the Maze (e.g., to do it in a 2-D manner?)
- Debugging
- Optional: figure out an algorithm to generate new maze