Computer Organization

Lab 0: Turing Machine

Lab Objective:

To understand the concept of computation by studying the operation of Turing Machines.

Introduction

A Turing Machine is a theoretical model of a computer. In this lab we will experiment with a Turing Machine simulator and write your own simple TMs to gain a better understanding of what computation is.

It is important to understand the mechanisms of a Turing Machine because there are a number of philosophical arguments that are based on this formalism. For example, the **Church-Turing thesis** is the hypothesis that anything that can be computed can be computed by a Turing Machine. In other words, **a Turing Machine captures all of the essential aspects of computation**.

A Turing Machine consists of:

- An infinite tape that is divided into squares, each of which can hold one character of information.
- A read/write head that can move to the left or right on the tape, one square each step.
- A program on a Turing Machine consists of a series of rules made up of five parts:
    1. Current state
    2. Current character being read
    3. Next state
    4. New Character to write
    5. Action to take (move to left or right)

The program of a TM can be considered to be **a transition function from (Current state, Current Character) to (Next State, New Character, Move Right/Left)**.

Lab Instruction

Step 1: Read about the Turing Machine Simulation webpage at

http://math.hws.edu/eck/js/turing-machine/TM-info.html

Step 2: Open the TM simulator at http://math.hws.edu/eck/js/turing-machine/TM.html

It shows an example Turing Machine that can add one to a binary number. A binary number (that is, a sequence of zeros and ones) must be written on the tape, and the Turing Machine must be positioned on the right end of that number.

Sample TM: The Binary Increment Turing machine will add one to a binary number and halt. It must be started on the right end of the number that is to be incremented.

```
{
  "name": "Binary Increment",
  "max_state": 25,
  "symbols": "xyzabc01$@",
  "tape": "1",
  "position": 0,
  "rules": [
    [ 0, "#", "1", 1, "R" ],
    [ 0, "0", "1", 1, "R" ],
    [ 0, "1", "0", 0, "L" ],
    [ 1, "#", "#", "h", "L" ],
    [ 1, "0", "0", 1, "R" ],
    [ 1, "1", "1", 1, "R" ]
  ]
}
```

Step 3: Click the "Run" button, and the machine will add one to the binary number. It will then return to its original position and halt.

Step 4: To run it again, first click the button that says "Reset State to Zero" to return the machine to its initial state, then click "Run". There is a "Run Speed" pop-up menu that allows you to change the delay between steps in the program.

Step 5: As alternative to "Run" you can click the "Step" button, which executes just one rule in the machines's program. There is also a "Step Back" button that will undo one step in the execution; you can use it to "unwind" part of a computation so that you can replay it. "Step Back" can undo up to 100 steps. You can use the mouse (or your finger on a touch screen) to drag the machine onto a different cell of the tape. You can also drag the tape back and forth, carrying the machine with it.

Step 6: Try to play with the example and run the TM on a variety of different inputs.

Step 7: Study the six rules of Binary Increment TM and try to understand what each rule does.

Lab Exercise:

1) **Write a Turing Machine program called Inverse that will read a tape containing eight bits of 1's and 0's and invert each number to the opposite number**. For example, if the input binary string is “11001111”, then the output should be “00110000”.

   Load your program to the TM simulator and run it and make sure it is correct. You can use "New Turing Machine" and the "Rule Editor" to create or modify Turing Machine programs. (Think: how many states does the TM need? How to start and how to stop? What are the rules?)

2) **Write a Turing Machine program that will count in binary**. Starting from an empty tape, it will count like this: “0 , 1, 10, 11, 100 ……” (Hint: you may study and modify the example code for count in base 10)
3) Use chatGPT or any generative AI tools to investigate whether **chatGPT is more powerful than the Universal Turing Machine** and write a summary on your findings. Please specify the AI tool used.

Submission

In ISpace, submit your Turing Machine programs along with your screens of running it on the TM simulator and your findings about question 3 before next Thursday. You may include everything into one single word file.

References:

http://morphett.info/turing/turing.html https://turingmachinesimulator.com/
http://courses.washington.edu/i300au02/labs/lab_4_using_machine.pdf
http://math.hws.edu/eck/js/turing-machine/TM.html