Programming Assignment 2

# Overview

The initial effort of this assignment was to implement a class structure that properly handled a deterministic turing machine as described in the assignment. This started with creating a DTM class that contained the six components: . From here, various supporting classes could be implemented to fully realize the initialization of a deterministic turing machine. These classes included the ControlRecord class, which held the symbol to write, direction to move, and next state for a given symbol, as well as the QState class, which held a dictionary of ControlRecord instances for the desired symbols that could be read, along with information about if the given state is a halting state.

After the class structure was implemented, the logic moving forward follows the process that any other turing machine would follow. That is, for a given state that is not a halting state, read the value of the tape at the current index, then, depending on the current state, write to the tape, move the read/write head, then move the current state as specified.

For each specific part, the same DTM class which can be initialized with different QState dictionaries, the dictionary for each part is what allows the turing machine to function the way it is intended.

# Part A

Part A was implemented using Table 1 specified in the assignment description.

# Part B

Part B utilizes a tape configured in a way that represents two different values, , separated by a blank symbol. An example would be if , then the input tape would be [1, 0, 1, 0, b, 1, 0, 0, 1]. The general process of all parts is to traverse over , traverse over , decrement , then do some sort of operation on , weather that be increment, decrement, or shift.

## i) Addition

Create a turing machine that computes the addition of two numbers. As specified above, this can be done by iterating over the process of decrementing and incrementing , until . This effectively computes .

This process is described in Table 1.

## ii) Subtraction

Create a turing machine that computes the subtraction of two numbers. As specified above, this can be done by iterating over the process of decrementing and decrementing , until . This effectively computes .

This process is described in Table 2.

This turing machine can only handle values such that . In the cases that , the turing machine exists with final tape of all zeros.

## iii) Multiplication (Extra Credit)

Create a turing machine that computes the multiplication of two numbers. As specified above, this can be done by iterating over the process of decrementing and logical shifting left , until . This effectively computes .

This process is described in Table 3.

A special case has been made to logical shift right one time at the end before halting, this is because multiplying by a number requires one less shift than the number. For example, the tape [1, 1, b, 1, 0] would result in the multiplication of , which would only require logical shift left, which is the same as logical shifts left and logical shift right.

Note: This does not fully work for all values of , it instead needs to be a factor of . For example, I was not able to fully realize all values for .

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | | | **1** | | | **B** | | |  |
|  | **State** | **Bit** | **Direction** | **State** | **Bit** | **Direction** | **State** | **Bit** | **Direction** |  |
| **Q0** | Q0 | 0 | R | Q0 | 1 | R | Q1 | B | R | Traverse M |
| **Q1** | Q0 | 0 | R | Q0 | 1 | R | Q2 | B | L | Traverse N |
| **Q2** | Q2 | 0 | L | Q3 | 0 | R | Q8 | B | L | Decrement N, Cleanup If Blank (N = 0) |
| **Q3** | Q4 | 1 | R | X | X | X | Q4 | B | L | Decrement N (Handle Borrow) |
| **Q4** | Q4 | 0 | L | Q4 | 1 | L | Q5 | B | L | Traverse Back Over N |
| **Q5** | Q0 | 1 | R | Q6 | 0 | L | Q0 | 1 | R | Increment M |
| **Q6** | Q0 | 1 | R | Q6 | 0 | L | Q0 | 1 | R | Increment M (Handle Carry) |
| **Q8** | Q8 | B | R | Q8 | B | R | QY | B | X | Cleanup |
| **QY** |  |  |  |  |  |  |  |  |  |  |

Table 1: States For Addition Turing Machine

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | | | **1** | | | **B** | | |  |
|  | **State** | **Bit** | **Direction** | **State** | **Bit** | **Direction** | **State** | **Bit** | **Direction** |  |
| **Q0** | Q0 | 0 | R | Q0 | 1 | R | Q1 | B | R | Traverse M |
| **Q1** | Q0 | 0 | R | Q0 | 1 | R | Q2 | B | L | Traverse N |
| **Q2** | Q2 | 0 | L | Q3 | 0 | R | Q8 | B | L | Decrement N, Cleanup If Blank (N = 0) |
| **Q3** | Q4 | 1 | R | X | X | X | Q4 | B | L | Decrement N (Handle Borrow) |
| **Q4** | Q4 | 0 | L | Q4 | 1 | L | Q5 | B | L | Traverse Back Over N |
| **Q5** | Q5 | 0 | L | Q6 | 0 | R | Q7 | B | R | Decrease M, Special Case If Blank (M = 0, M < N) |
| **Q6** | Q6 | 1 | R | X | X | X | Q1 | B | R | Decrease M, Handle Borrow |
| **Q7** | Q7 | 0 | R | Q7 | 0 | R | Q8 | B | R | Clear N When M < N |
| **Q8** | Q8 | B | R | Q8 | B | R | QY | B | X | Cleanup |
| **QY** |  |  |  |  |  |  |  |  |  |  |

Table 2: States For Subtraction Turing Machine

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | | | **1** | | | **B** | | |  |
|  | **State** | **Bit** | **Direction** | **State** | **Bit** | **Direction** | **State** | **Bit** | **Direction** |  |
| **Q0** | Q0 | 0 | R | Q0 | 1 | R | Q1 | B | R | Traverse M |
| **Q1** | Q0 | 0 | R | Q0 | 1 | R | Q2 | B | L | Traverse N |
| **Q2** | Q2 | 0 | L | Q3 | 0 | R | Q9 | B | L | Decrement N, Cleanup If Blank (N = 0) |
| **Q3** | Q4 | 1 | R | X | X | X | Q4 | B | L | Decrement N (Handle Borrow) |
| **Q4** | Q4 | 0 | L | Q4 | 1 | L | Q5 | B | L | Traverse Back Over N |
| **Q5** | Q5 | 0 | L | Q6 | 0 | L | Q0 | 0 | R | Shift 0 |
| **Q6** | Q5 | 1 | L | Q5 | 1 | L | Q0 | 1 | R | Shift 1 |
| **Q7** | Q7 | 0 | L | Q7 | 1 | L | Q8 | B | R | Traverse Back Over M |
| **Q8** | Q8 | 0 | R | Q9 | 0 | R | Q10 | B | X | Shift 0 Right |
| **Q9** | Q8 | 1 | R | Q9 | 1 | R | Q10 | B | X | Shift 1 Right |
| **Q10** | Q8 | B | R | Q8 | B | R | QY | B | X | Cleanup |
| **QY** |  |  |  |  |  |  |  |  |  |  |

Table 3: States For Multiplication Turing Machine