Report on Tomasulo's Algorithm Simulator

Implementation Overview

This program simulates Tomasulo's algorithm, a hardware algorithm for dynamic instruction scheduling.

Key features include:

Dynamic Reservation Stations Configuration: Users can specify the number of reservation stations for each functional unit and the number of cycles each operation takes.

Graphical User Interface (GUI): Developed using Tkinter, the GUI allows users to add instructions, load them from a file, and run/reset the simulation.

Cycle-by-Cycle Simulation(In terminal): The program tracks and displays the issue, execute, and write-back times for each instruction.

User Guide

Initial Setup

1. **Launch the Program:** Start the program to open the initial setup window.
2. **Configure Settings:** Specify the number of reservation stations and cycles needed for each functional unit type.
3. **Apply Settings**: Click "Apply" to save the settings and open the main simulation window.
4. Adding Instructions

5. **Select Operation:** Choose the operation from the dropdown menu.
6. **Enter Operands:** Fill in the relevant fields for the selected operation.
7. **Add Instruction:** Click "Add Instruction" to add it to the list.
8. Loading Instructions from a File
9. **Load File:** Click "Load Instructions" and select a file containing the instructions.
10. **Display Instructions:** The instructions will be loaded and displayed in the list.
11. Running the Simulation
12. **Run Simulation:** Click "Run Simulation" to start the cycle-by-cycle execution.
13. **View Results:** The simulation results, including cycle count and register values, are displayed.
14. Resetting the Simulation
15. **Reset Simulation:** Click "Reset Simulation" to clear instructions and reset the system state.

Full Simulation Example

Step-by-Step Example

Initial Setup:

Configure the following settings:

LOAD: 2 reservation stations, 6 cycles

STORE: 1 reservation station, 6 cycles

ADD: 4 reservation stations, 2 cycles

NAND: 2 reservation stations, 1 cycle

MUL: 1 reservation station, 8 cycles

BEQ: 1 reservation station, 1 cycle

CALL: 1 reservation station, 1 cycle

RET: 1 reservation station, 1 cycle

ADDI: 2 reservation stations, 2 cycles

Adding Instructions:

LOAD R6, 1(R0)

LOAD R2, 2(R0)

ADD R3, R6, R2

STORE R3, 6(R0)

NAND R4, R6, R2

MUL R5, R6, R2

ADDI R6, R6, 60

BEQ R0, R0, 2

ADDI R7, R0, 20

ADDI R7, R0, 21

ADDI R7, R0, 20

ADDI R3, R0, 40

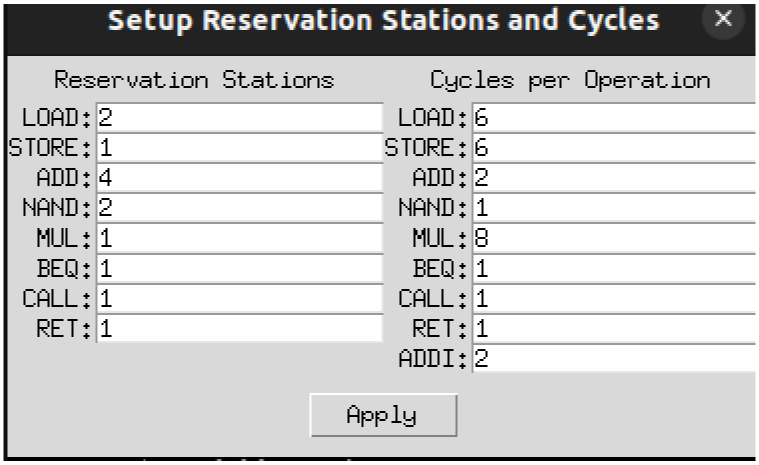
Running the Simulation:

Click "Run Simulation" to start.

The cycle-by-cycle results will display issue, execute, and write-back times for each instruction.

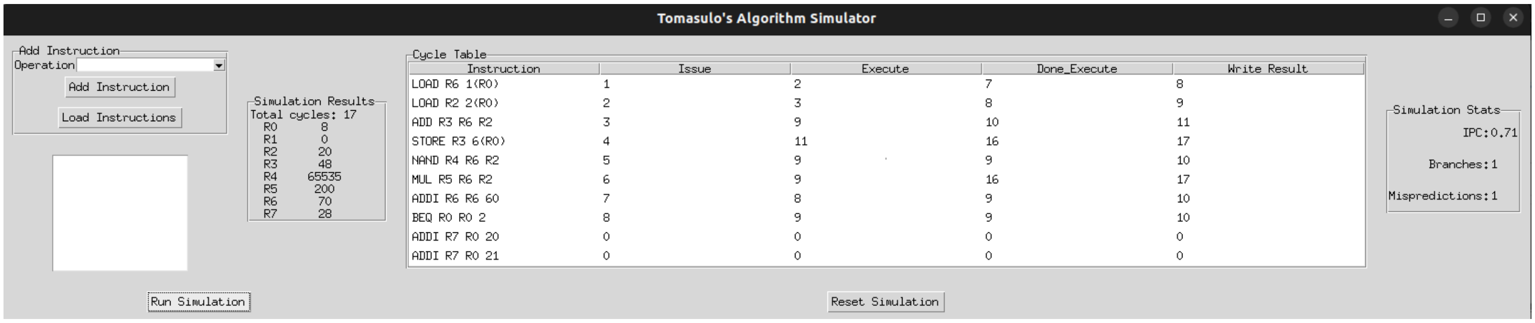
Simulation Snapshots

Setup Screen:



	Reservation Stations		Cycles per Operation
LOAD:	2	LOAD:	6
STORE:	1	STORE:	6
ADD:	4	ADD:	2
NAND:	2	NAND:	1
MUL:	1	MUL:	8
BEQ:	1	BEQ:	1
CALL:	1	CALL:	1
RET:	1	RET:	1
		ADDI:	2

Tomasulo Full View:



Simulation Results

```
Total cycles: 17
  R0       8
  R1       0
  R2      20
  R3      48
  R4   65535
  R5     200
  R6      70
  R7      28
```

Cycle Table

Instruction	Issue	Execute	Done_Execute	Write Result
LOAD R6 1(R0)	1	2	7	8
LOAD R2 2(R0)	2	3	8	9
ADD R3 R6 R2	3	9	10	11
STORE R3 6(R0)	4	11	16	17
NAND R4 R6 R2	5	9	9	10
MUL R5 R6 R2	6	9	16	17
ADDI R6 R6 60	7	8	9	10
BEQ R0 R0 2	8	9	9	10
ADDI R7 R0 20	0	0	0	0
ADDI R7 R0 21	0	0	0	0

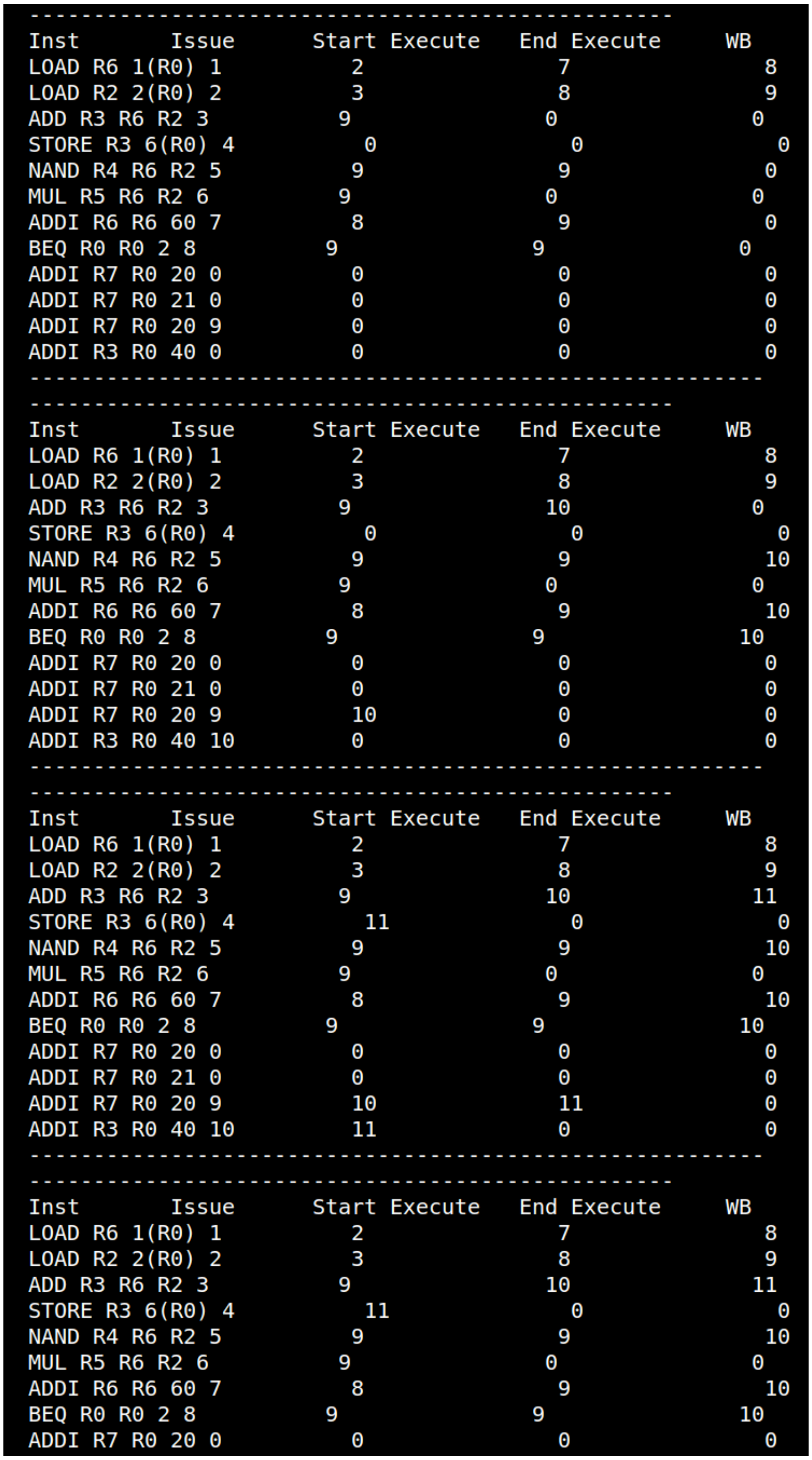
Simulation Stats

IPC:0.71

Branches:1

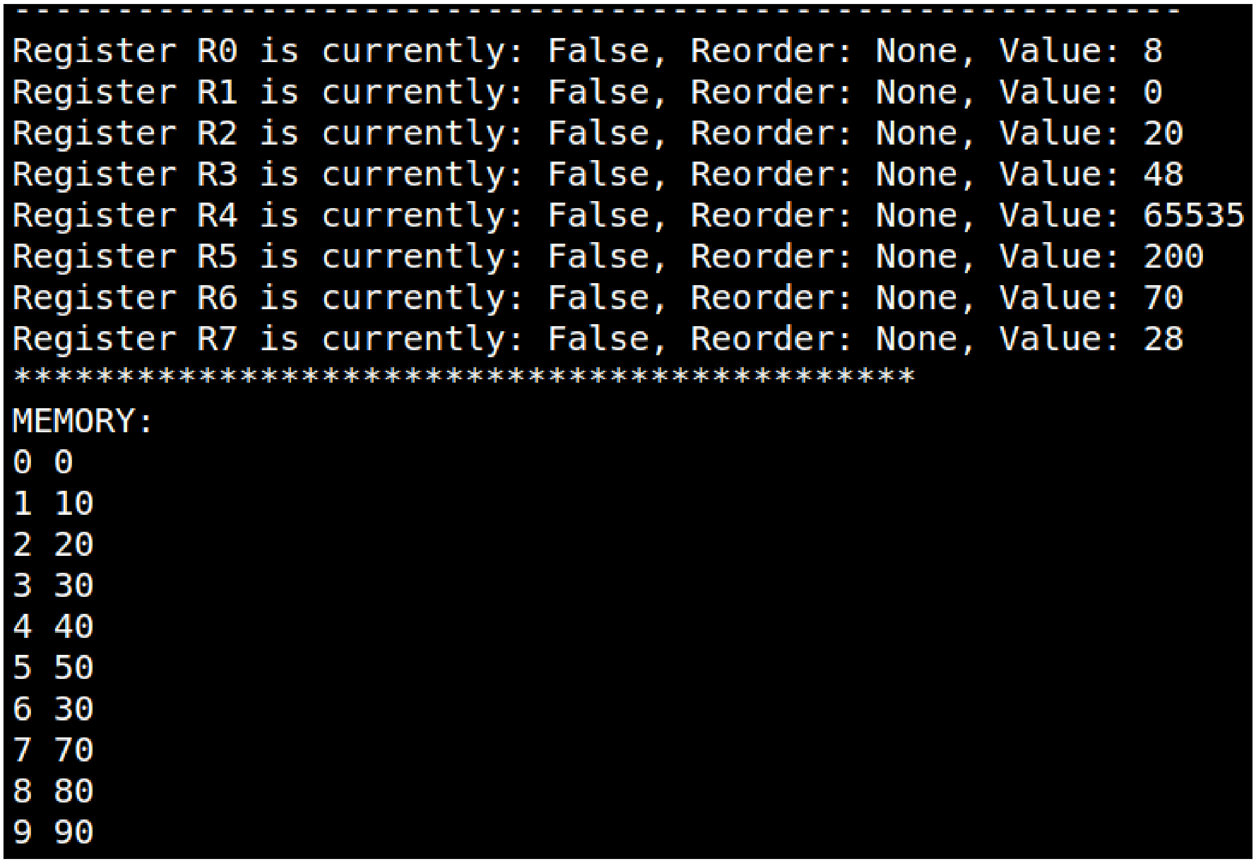
Mispredictions:1

Cycle by cycle terminal view:



```
--------------------------------------------------
Inst       Issue      Start Execute   End Execute     WB
LOAD R6 1(R0) 1          2               7               8
LOAD R2 2(R0) 2          3               8               9
ADD R3 R6 R2 3          9               0               0
STORE R3 6(R0) 4          0               0               0
NAND R4 R6 R2 5          9               9               0
MUL R5 R6 R2 6          9               0               0
ADDI R6 R6 60 7          8               9               0
BEQ R0 R0 2 8          9               9               0
ADDI R7 R0 20 0          0               0               0
ADDI R7 R0 21 0          0               0               0
ADDI R7 R0 20 9          0               0               0
ADDI R3 R0 40 0          0               0               0
---------------------------------------------------------
--------------------------------------------------
Inst       Issue      Start Execute   End Execute     WB
LOAD R6 1(R0) 1          2               7               8
LOAD R2 2(R0) 2          3               8               9
ADD R3 R6 R2 3          9               10              0
STORE R3 6(R0) 4          0               0               0
NAND R4 R6 R2 5          9               9               10
MUL R5 R6 R2 6          9               0               0
ADDI R6 R6 60 7          8               9               10
BEQ R0 R0 2 8          9               9               10
ADDI R7 R0 20 0          0               0               0
ADDI R7 R0 21 0          0               0               0
ADDI R7 R0 20 9          10              0               0
ADDI R3 R0 40 10         0               0               0
---------------------------------------------------------
--------------------------------------------------
Inst       Issue      Start Execute   End Execute     WB
LOAD R6 1(R0) 1          2               7               8
LOAD R2 2(R0) 2          3               8               9
ADD R3 R6 R2 3          9               10              11
STORE R3 6(R0) 4          11              0               0
NAND R4 R6 R2 5          9               9               10
MUL R5 R6 R2 6          9               0               0
ADDI R6 R6 60 7          8               9               10
BEQ R0 R0 2 8          9               9               10
ADDI R7 R0 20 0          0               0               0
ADDI R7 R0 21 0          0               0               0
ADDI R7 R0 20 9          10              11              0
ADDI R3 R0 40 10         11              0               0
---------------------------------------------------------
--------------------------------------------------
Inst       Issue      Start Execute   End Execute     WB
LOAD R6 1(R0) 1          2               7               8
LOAD R2 2(R0) 2          3               8               9
ADD R3 R6 R2 3          9               10              11
STORE R3 6(R0) 4          11              0               0
NAND R4 R6 R2 5          9               9               10
MUL R5 R6 R2 6          9               0               0
ADDI R6 R6 60 7          8               9               10
BEQ R0 R0 2 8          9               9               10
ADDI R7 R0 20 0          0               0               0
```

Final Memory State:



```
---------------------------------------------------------
Register R0 is currently: False, Reorder: None, Value: 8
Register R1 is currently: False, Reorder: None, Value: 0
Register R2 is currently: False, Reorder: None, Value: 20
Register R3 is currently: False, Reorder: None, Value: 48
Register R4 is currently: False, Reorder: None, Value: 65535
Register R5 is currently: False, Reorder: None, Value: 200
Register R6 is currently: False, Reorder: None, Value: 70
Register R7 is currently: False, Reorder: None, Value: 28
********************************************
MEMORY:
0 0
1 10
2 20
3 30
4 40
5 50
6 30
7 70
8 80
9 90
```

Discussion of Results

The simulation successfully demonstrates the functionality of Tomasulo's algorithm. Key observations include:

Instruction Overlap: Multiple instructions were issued and executed concurrently, illustrating out-of-order execution.

Resource Allocation: The specified reservation stations and cycles ensured efficient resource utilization.

Branch Handling: The BEQ instruction correctly updated the program counter, demonstrating branch handling.

Overall, the program effectively simulates Tomasulo's algorithm, providing valuable insights into dynamic instruction scheduling and out-of-order execution.