

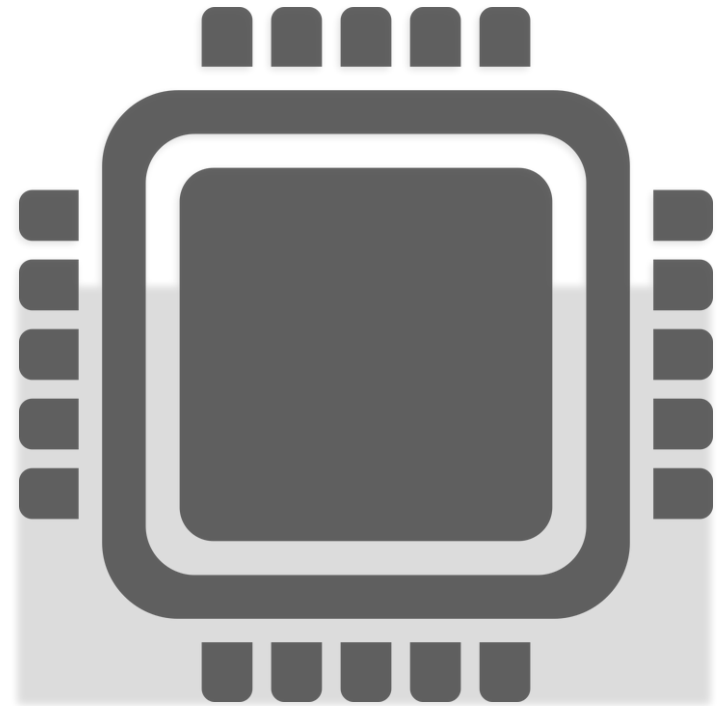
# MIPS with Tomasulo Simulator

ECE-668 Semester Project

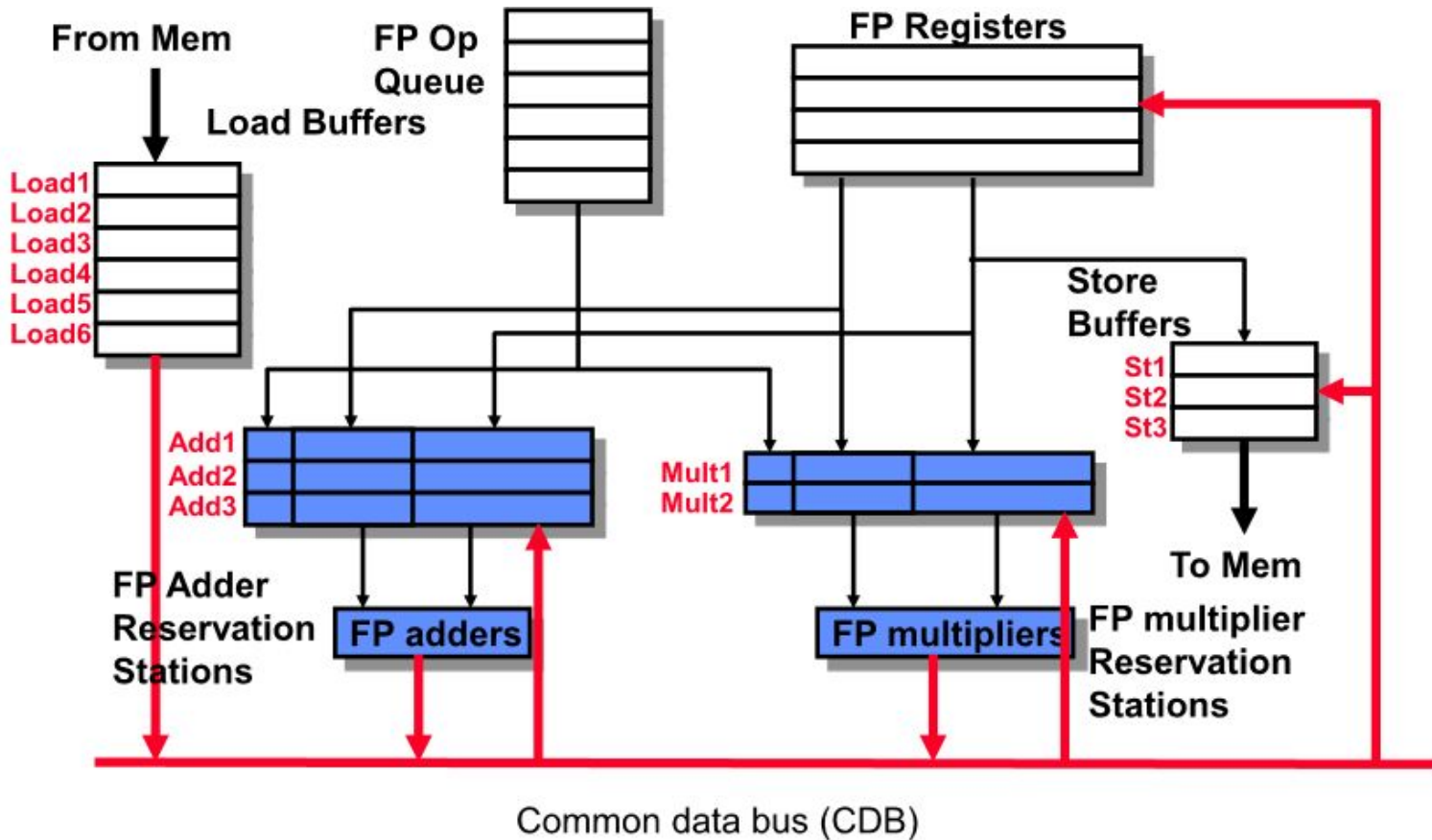
**Presented by:** Omar Areiqat  
Raja Hasnain Anwar  
Shayan Nazeer

# Introduction

- Tomasulo Algorithm is a **hardware approach** for the **dynamic scheduling**.
- It allows **out of order** execution.
- By tracking operands for instructions in the reservation stations and register renaming in hardware the algorithm minimizes RAW and eliminates WAW and WAR hazards.



# The Algorithm



[https://www.google.com/url?sa=i&url=https%3A%2F%2Fleesangwon0114.github.io%2Fcomputerarchitecture%2F2018%2F06%2F02%2FComputerArchitecture\\_10.Tomasulo-Algorithm-and-Dynamic-Branch-Prediction.html&psig=AOvVaw1q7A\\_T3IRB0inEGNc8yMN6&ust=1701804118114000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCIjchKTB9oIDFQAAAAAdAAAAABAI](https://www.google.com/url?sa=i&url=https%3A%2F%2Fleesangwon0114.github.io%2Fcomputerarchitecture%2F2018%2F06%2F02%2FComputerArchitecture_10.Tomasulo-Algorithm-and-Dynamic-Branch-Prediction.html&psig=AOvVaw1q7A_T3IRB0inEGNc8yMN6&ust=1701804118114000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCIjchKTB9oIDFQAAAAAdAAAAABAI)

# The Simulator

---

We developed the **software simulator for the tomasulo's algorithm** with following **features**:

- 1) MIPS instruction scheduling.
- 2) Support for immediate instructions.
- 3) Additional support for Branch and MEM operations.
- 4) A GUI (Graphical User Interface).

## ISA Subset

The simulator is designed to handle following instructions:

ADD SUB MUL DIV	DADDI DSUBI MULI DIVI	LOAD STORE	BRANCH BEQZ BNEQZ
--------------------------	--------------------------------	---------------	-------------------------

# Overview

- We implemented OOP-based architecture to map different stages of the simulator.
- Our program works in two-phases:
  - First phase is the scheduling and program execution.
  - Second phase is visualization of scheduling step-by-step.
- The code is divided into 4 classes:
  - “**Instruction**” class parses the instruction and set instruction index.
  - “**Register**” class represents the register in the processor.
  - “**Reservation Station**” class represents the reservation station units in algorithm.
  - “**Reservation station manager**” is the backbone of the scheduling.

# Overview: Execution

```

337  # Execution times and station counts for different instruction types
338  execution_times = {"ADD": 4, "MUL": 1, "DIV": 40, "STORE": 3, "BRANCH": 3}
339  station_counts = {"ADD": 2, "MUL": 2, "DIV": 1, "STORE": 2, "BRANCH": 1}
340  optypes = { "ADD": ["ADD", "SUB", "DADDI", "DSUBI"],
341             "MUL": ["MUL", "MULI"],
342             "DIV": ["DIV", "DIVI"],
343             "STORE": ["STORE", "LOAD"],
344             "BRANCH": ["BRANCH", "BEQZ", "BNEQZ"]}

346  # Initialize the Reservation Station Manager with initial register values
347  initial_values = ["R1 80", "R2 10", "R3 80", "R4 13", "R5 14", "R6 15", "R7 16",
348                  "M0 8", "M8 16", "M16 32", "M24 64", "M32 128"]

# List of instructions to be issued
# instructions = ["SUB R1, R2, R3", "ADD R1, R2, R3", "ADD R7, R3, R3", "ADD R2, R7,
# instructions = ["ADD R1, R1, R3", "ADD R4, R5, R3", "DSUBI R10, R1, #100"]
# instructions = ["ADD R1, R1, R3", "ADD R4, R5, R3", "DSUBI R10, R1, #100", "BNEQZ
instructions = ["ADD R1, R1, R2", "BNEQZ R10, 0, X", "LOAD R1, M8, X"]

```

# Overview

Control Buttons & current clock cycle

Previous

Next

Current Cycle: 23

Reset

Instruction QUEUE

Instr	Name	Stage	Cycles Left	Src1	Src2	Dest	Vj	Vk	Qj	Qk
ADD R4 R8 R7	ADD_1	Execute	3	R8	R7	R4	0	0	None	None
ADD F1 F2 F3	ADD_2	Execute	4	F2	F3	F1	5.4	0.0	None	None
MUL_1	Idle	0								
MUL_2	Idle	0								
DIV_1	Idle	0								
STORE_1	Idle	0								
STORE_2	Idle	0								
BRANCH_1	Idle	0								

Register File

Int Register	Value
R1	10
R2	11
R3	12
R4	13
R6	0
R7	0
R8	0

Memory Support

Memory Value	Value
M8	16



# Overview: RSM

- Reservation Station Manger (RSM) implements the main scheduling algorithm in the simulator.
- It maintains:
  - Instruction Queues
  - Execution Cycles
  - Reservation Stations
  - Read/Write Restrictions (locks)
- It evaluations:
  - Data Dependencies and **Hazards**
    - RAW, WAR, WAW
  - Stalls
  - Instruction Execution
  - Register Updates

# Branch Instructions

- Two types of branching:
  - Conditional: BRANCH
  - Unconditional: BEQZ, BNEQZ
- Instead of branch labels, we branch to the Instruction Number.
  - E.g., *BNEQZ R10 0 X* loops to 0th Instruction if R10 value is NOT zero.
- No branch prediction, so we *stall* the pipeline until the branch is resolved.
- Branch is not evaluated until all previous instructions have finished.

STORE_2	Idle	0								
BNEQZ R10 0 X	BRANCH_1	Execute	1	R10	X	X	0	0	None	None

# Immediate Instructions

- Support for DADDI, DSUBI, MULI, DIVI.
- Immediate values are treated the same way as register values.
- Immediate source omitted for evaluating hazards and read/write restrictions (locks).

Previous	Next	Current Cycle: 8	Reset							
Instr	Name	Stage	Cycles Left	Src1	Src2	Dest	Vj	Vk	Qj	Qk
DADDI R1 R2 #3	ADD_1	Execute	4	R2	3	R1	11	3	None	None
ADD R7 R3 R3	ADD_2	Issue	5	R3	R3	R7	None	None	None	None

# Memory LD/SD

- Memory locations are predefined at the time of execution.
- Only involved at the time of loading and storing data.
  - Loads data into register.
  - Stores register data in memory locations.
- Practically work the same was as registers.
  - But not included in Hazard evaluation.


Memory Value	Value
M8	16

```


346 # Initialize the Reservation Station Manager with initial register values
347 initial_values = ["R1 80", "R2 10", "R3 80", "R4 13", "R5 14", "R6 15", "R7 16",
348                  "M0 8", "M8 16", "M16 32", "M24 64", "M32 128"]









```

# Team Collaboration

 MIPS\_tomasulo Private Unwatch 1

main 1 branch 0 tags Go to file Add file Code

 OAreiqat branches work fully 77633eb yesterday 12 commits

 .gitignore	added todos	last week
 Deliverables.pdf	init	last week
 README.md	Update display of cycle number	yesterday
 test.py	modified GUI to prep for mem, added couple functions to test.py	yesterday
 test_gui.py	accidentally removed cycle counter, fixed it	yesterday
 test_in_progress.py	branches work fully, theoretically	yesterday
 tomasulo_ALGO.py	init	last week
 tomasulo_GUI.py	init	last week

LIVE DEMO!!

Thank you for listening

University of  
Massachusetts  
Amherst

---