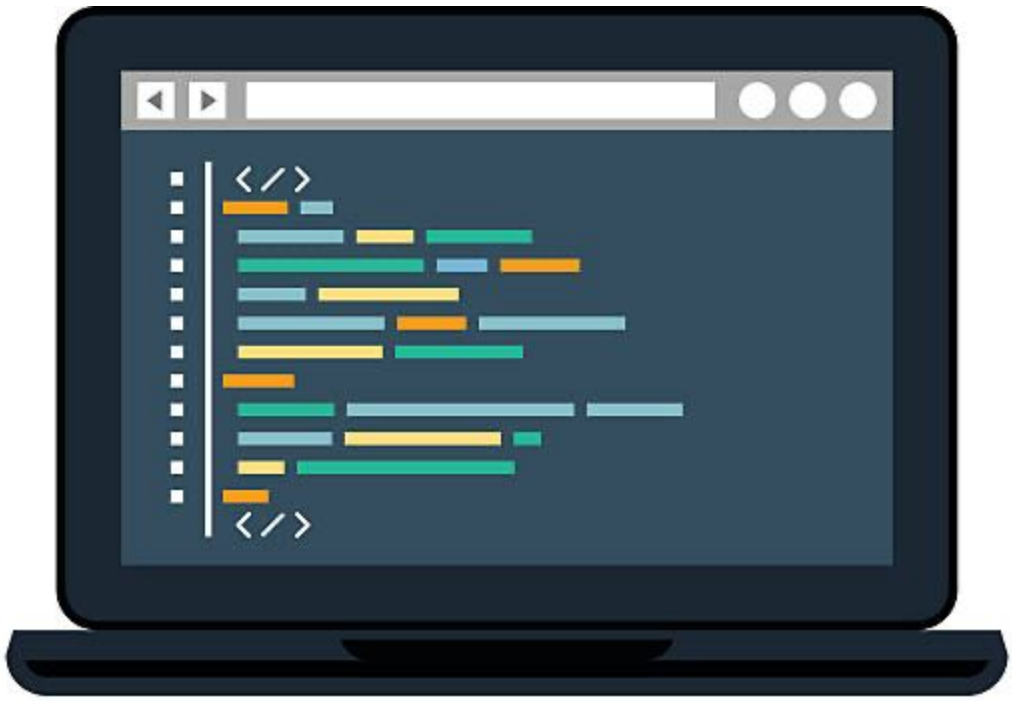
TOMASULO'S SIMULATION

Computer Architecture
Project 2



Ramy ElGendi - 900170269

Sara Ahmed - 900170870

Iman Yahia - 900171249

First we created 3 structs for the format of: Each instruction to be entered, the reservation station, and the register status.
Then, we created an input file to take the instructions in the following format: OPCODE RD RS1 RS2/IMM. And, another 2 output files, one to output the simulation with -1 and another one to remove the -1 (Final file).

We print 3 things each cycle: the registers status table, the ROB table, the reservation stations table, and the detailed instructions table (according to what we took on course slides).
At the end, a summary table is printed which describes at which cycle each instruction was issued,executed,written,and committed.

We created a function called issue,execute,commit and write to deal with the tomasulo's simulation with speculation and we called each 2 times in one clock cycle to demonstrate dual core.

Brief description of each function:

- checkfile : Make sure the file exists and is empty
- RemoveEmpty : Removes all Empty signs (-1) from text file
- Process_File : Reads instructions in the file and places them in a struct with special format called ins_format for each instruction, and fills in the necessary data for each instruction.
- Simulation : Creates the reservation station and sets execution clock cycle for each instruction, creates registers and loads information intro the dynamic memory created. Also. begins the simulation by calling the 4 functions (twice).
- PrintTable1 : Print each cycle
- PrintTable2 : Print cycles summary and branch predictions.
- issue,execute,writeResult : The functions needed for Tomasulo’s simulation. Simple perfect case: First it issues the instruction, then execute its in the clock cycles given before and writes it back then commits it.

Console Output:

```
List of instructions uploaded forom assembly_instructions.txt:

      Instruction 0 → ADD 1 1 1
      Instruction 1 → MULT 1 1 1
      Instruction 2 → ADD 1 1 2
      Instruction 3 → SW 1 1 2
      Instruction 4 → LW 1 1 2
      Instruction 5 → NAND 3 1 2
      Instruction 6 → ADDI 1 3 2
      Instruction 7 → BEQ 1 1 2
      Instruction 8 → JMP -1 -1 2
      Instruction 9 → JALR -1 2 -1
      Instruction 10 → RET -1 -1 -1

• There is a total of [11] instructions uploaded from the file.

✔ File: simulation_results.txt created successfuly! ✔

- Results:
      Result of Instruction 0 is 2.
      Result of Instruction 1 is 1.
      Result of Instruction 2 is 3.
      Result of Instruction 3 is 3.
      Result of Instruction 4 is 3.
      Result of Instruction 5 is 0.
      Result of Instruction 6 is 1.
      Result of Instruction 7 is 1.27618e-228.
      Result of Instruction 8 is 13.
      Result of Instruction 9 is 2.
      Result of Instruction 10 is 1.

- Simulation Completed!

Program ended with exit code: 0
```

PrintTable2 Output:

Summary

Ins	Issue	Ex start	Ex end	Write	Committed
ADD	1	2	2	3	4
MULT	1	2	6	7	8
ADD	2	3	3	4	5
SW	2	3	4	5	6
LW	3	4	5	6	7
NAND	3	4	4	5	6
ADDI	4	5	5	8	9
BEQ	4	5	5	6	7
JMP	5	6	6	7	8
JALR	6	7	7	8	9
RET	6	7	7	9	10

The misprediction percentange is: nan

How each cycle is saved in the text file:simulation_results_2:

```
  CYCLE 7
------------
  Started executing instruction 10
  Finished executing instruction 10
  Started executing instruction 11
  Finished executing instruction 11
---------------------------------------------------------------------
    Name      |  Busy |    Op |    Vj |    Vk |    Qj |    Qk | Address
---------------------------------------------------------------------
ADD/SUB/ADDI1 |     0 |       |       |       |       |       | 0+R-1
ADD/SUB/ADDI2 |     0 |       |       |       |       |       | 0+R-1
ADD/SUB/ADDI3 |     0 |       |       |       |       |       | 0+R-1
        MULT1 |     0 |       |       |       |       |       | 0+R-1
        MULT2 |     0 |       |       |       |       |       | 0+R-1
          LW1 |     0 |       |       |       |       |       | 0+R-1
          LW2 |     0 |       |       |       |       |       | 0+R-1
          SW1 |     0 |       |       |       |       |       | 0+R-1
          SW2 |     0 |       |       |       |       |       | 0+R-1
        NAND1 |     0 |       |       |       |       |       | 0+R-1
        NAND2 |     0 |       |       |       |       |       | 0+R-1
         JMP1 |     0 |       |       |       |       |       | 0+R-1
         JMP2 |     0 |       |       |       |       |       | 0+R-1
        JALR1 |     1 |  JALR |     2 |     0 |       |       | 0+R-1
        JALR2 |     1 |  ADDI |     0 |     2 |       |       | 0+R-1
         RET1 |     0 |       |       |       |       |       | 0+R-1
         RET2 |     1 |   RET |     0 |     0 |       |       | 0+R-1
        ADDI1 |     0 |       |       |       |       |       | 0+R-1
        ADDI2 |     0 |       |       |       |       |       | 0+R-1
        ADDI3 |     0 |       |       |       |       |       | 0+R-1
         BEQ1 |     0 |       |       |       |       |       | 0+R-1
         BEQ2 |     0 |       |       |       |       |       | 0+R-1
=====================================================================
----------------
REGISTER STATUS
----------------
0  | Qi =
1  | Qi =
2  | Qi =
3  | Qi =
4  | Qi =
5  | Qi =
6  | Qi =
7  | Qi =
================
----------------------------------------
```

```
----------------------------------------
            ROB
----------------------------------------
    #    |  Type |  Dest | Value | Ready |
----------------------------------------
    1    |       |       |       |       |
    2    |       |       |       |       |
    3    |       |       |       |       |
    4    |       |       |       |       |
    5    |       |       |       |       |
    6    |       |       |       |       |
    1    | ADDI  |     1 |       |    NO |
    8    |       |       |       |       |
------------
```